

LOCKHEED MISSILES & SPACE COMPANY
HUNTSVILLE RESEARCH & ENGINEERING CENTER
HUNTSVILLE RESEARCH PARK
4800 BRADFORD DRIVE, HUNTSVILLE, ALABAMA

FINAL REPORT
SOLUTIONS OF SYSTEMS OF
NONLINEAR EQUATIONS

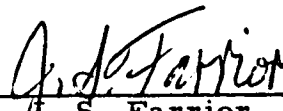
28 October 1966

Contract NAS8-20178

APPROVED BY:



G. E. Christopher
Mgr., Dynamics & Guidance



J. S. Farrior
Resident Manager

FOREWORD

This report documents and summarizes work accomplished in the Solutions of Systems of Nonlinear Equations, Contract NAS8-20178. It includes a complete discussion of the theory, a bibliography of the literature consulted during the study, a user's manual and a programmer's manual of the resulting computer programs.

This work was performed by Lockheed Missiles & Space Company, Huntsville Research & Engineering Center, for the Aero-Astrodynamic Laboratory of the George C. Marshall Space Flight Center. Contributors to this study were K. L. Remmler, D. W. Cawood, J. A. Stanton, and R. Hill.

SUMMARY

This report presents the results of a study to develop a method and computer program for solving an arbitrary, simultaneous system of nonlinear algebraic and transcendental equations. A review of the literature and the theory regarding some particular methods (Newton-Raphson, False Position, Fletcher-Powell, Simplex Search, Sequential Minimax Search, and Contour Mapping) are discussed. Computation schemes for digital computer facilities are emphasized, however, the feasibility and attractive features of hybrid computation schemes are also discussed. The final result of this study is a composite computer program which encompasses a "limited" spectrum of basically different numerical methods - gradient, minimization, and search techniques. Test results are included as a basis for comparison of the different methods.

ACKNOWLEDGEMENT

The authors are particularly grateful for the cooperation, suggestions and technical direction provided by R. S. Ryan of the NASA/MSFC Aero-Astroynamics Laboratory.

CONTENTS

Section		Page
	FOREWORD	ii
	SUMMARY	iii
	ACKNOWLEDGEMENT	iv
1	INTRODUCTION	1
2	TECHNICAL DISCUSSION	2
	2.1 A General Survey of Methods for Solving Nonlinear Equations	3
	2.2 Review of Some Particular Methods	6
	2.3 Analog and Hybrid Computation Schemes	33
	2.4 Test Results and Engineering Applications	42
3	RECOMMENDATIONS AND CONCLUSIONS	75
4	REFERENCES	78
Appendix		
A	BIBLIOGRAPHY	A-1
B-1	COMPOSITE PROGRAM USER'S MANUAL	B1-1
B-2	FALSE-POSITION PROGRAM USER'S MANUAL	B2-1
C-1	PROGRAMMER'S MANUAL	C1-1
C-2	FALSE-POSITION PROGRAM LISTING	C2-1

Section 1

INTRODUCTION

The analysis of nonlinear systems has become a necessity in numerous problem areas associated with the development of aerospace vehicles. Consequently, equations requiring solution are quite varied and seldom are of a form for which a known solution exists.

One particular problem area of interest is the analysis of nonlinear dynamical system in which the solution of nonlinear differential equations is attempted by the Ritz Averaging Method. This procedure is discussed in References 1 and 2. In applying the Ritz Averaging Method, one is confronted by the problem of determining the solution to a simultaneous system of nonlinear algebraic and transcendental equations. The dynamical systems to which this technique can be applied are presumably quite numerous, however, specific examples for which it has proven to be useful are nonlinear vibration and control system problems. Due to the complexity and peculiar characteristics usually associated with nonlinear dynamical systems, a capability for solving widely diverse classes of nonlinear equations is an essential requirement.

There are many methods that have been formulated for solving nonlinear equations and the variations or modifications to these methods are numerous. One very logical explanation for the existence of so many methods and variations thereof is simply that any one method is frequently not suited to a particular system of equations.

Section 2

TECHNICAL DISCUSSION

An important by-product of this study contract has been the knowledge, insight, and notions gained through a thorough review of the literature on methods for solving nonlinear equations. The first subsection of this discussion is therefore devoted to a brief survey of the numerous methods and variations thereof that had been discovered in the literature. Many of the methods included in this survey were developed for minimization problems; therefore the authors may not have had their application to solving nonlinear equations in mind. They are indeed applicable, however, and offer some very good approaches to the problem at hand.

The essential purpose of the literature survey was to find the techniques, or combination of techniques that would provide a basis for accomplishing the contractual objective - the objective being a computer program for solving an arbitrary simultaneous system of nonlinear algebraic and transcendental equations. No one method appeared to be superior than all others in being completely general regarding its application to arbitrary equations, the selection of a particular method being dictated by the particular system of equations. Hence a combination of methods encompassing to a "limited" extent the complete spectrum of basically different methods, was chosen as the best approach.

The major factors influencing the selection of these particular methods were

1. simplicity of the logic and computation scheme
2. reliability regarding convergence
3. accuracy
4. popularity

The significance of these factors is obvious, however, the "yardstick" for measuring them is nebulous and often arbitrary. Therefore some equally good (possibly better) techniques were excluded solely to preclude an unwieldy computer program. Some specific comments relating to these factors are included in Section 2.1. A detailed discussion of the theory for particular techniques selected is provided in Section 2.2.

Another major factor which generally influences the choice of a particular method is the particular computer facility available - whether it be analog, hybrid, or digital, as well as the size and speed limitations. The facility for which the present method has been developed is the NASA/MSFC Computation Laboratory's IBM 7094. Since some techniques appear to be especially attractive for repetitive analog and hybrid computation, a general discussion of such computation schemes is included in Section 2.3.

Generality of the computer program has been emphasized. However, the requirement for such a program is strengthened by the existence of particular applications. Section 2.4 provides a typical example of an engineering problem whose solution demands an efficient technique such as the one developed in this study.

2.1 A GENERAL SURVEY OF METHODS FOR SOLVING NONLINEAR EQUATIONS

A bibliography of selected articles consulted in the performance of this study is provided as Appendix A of this report. This is not a complete list of all the literature available on the subject; however, it is an attempt to include the most significant work representing a complete spectrum of different approaches and major variations thereof. A basic division of the techniques might be as follows: (1) gradient methods, (2) direct search, and (3) random search. One approach frequently employed, which can be accomplished by techniques classified in any one of these three categories, is to replace the system of equations by the problem of finding the minimum of a single function. The Fletcher-Powell and Simplex Methods are examples of this approach. All

three of the basic techniques are amenable to both digital, and hybrid computers. In general, however, the search methods appear more attractive when repetitive analog and hybrid computers are available.

Some general references such as textbooks and survey papers, as well as articles concerning specific techniques, are included in this bibliography. Brooks (5), Freudenstein (15), Hochstrasser (17), Householder (19), Levine (24), Ostrowski (32), and Saaty (36) describe fundamental theory and give a general survey of classical methods. These include for example the Newton-Raphson Method, Regula-Falsi, Graeffe Method, steepest descent, relaxation methods, and random search methods. Spang (38) provides the most comprehensive survey and complete review. Brooks (5) attempts to compare various methods on an experimental basis.

Specific gradient techniques are presented by Barnes (1), Booth (3), Broyden (6), Crockett (7), Curry (8), Davidon (9), Fletcher (13) and (14), Kizner (23), Powell (33) and (34), Rosenbrock (35), Shah (37), and Wolfe (41). The method proposed by Broyden (6), is a modification to the Newton-Raphson technique which would presumably result in a savings of computer time with some loss in accuracy. Since the computer time of the Newton-Raphson routine in the present program appears to be negligible, this modification was not employed. Gradient methods appear to be the most abundant in the literature and are employed quite widely in the aerospace industry for trajectory and system optimization studies. The three most popular methods for solving systems of algebraic equations appear to be the Newton-Raphson, False Position, and Fletcher-Powell methods. The Fletcher-Powell method is however, rather new and hence not nearly as proven a technique as many others. Therefore its selection was not without some anxiety. The final results obtained with this program, however, were in general, better than those obtained from any of the other methods.

Various search strategies are presented by Berman (2), Himsworth (16), Hooke (18), Johnson (20), Kiefer (21), Fletcher (12) and (22), Nelder (30), and

and Swann (39). An advantage of search methods is that they do not require the computation of a gradient; hence, regularity and continuity conditions (such as the existence of derivatives) are not required of the function to be minimized. If the gradient of a function cannot be determined analytically or by a finite difference approximation, a search technique is necessary. Johnson and Kiefer present a sequential procedure, which involves Fibonacci numbers, for locating the minimum of a unimodal function. This is an optimal procedure in that any other sequential search may require a larger number of evaluations. Berman presents a family of procedures which is simpler than that proposed by Johnson and Kiefer, and makes the claim that these procedures require about the same number of evaluations as the Fibonacci method. Himsworth and Nelder discuss the application of a sequential search using a simplex. The simplex method proposed by Nelder and Mead is included in the composite computer program and a complete description of this method is given in the sequel. This particular method was selected because of its simplicity and the immediate success that was achieved in its application to typical problems. Hooke presents a search technique which combines the aspects of both the gradient and univariate search techniques. After each point, a univariate search is made around that point to determine the direction to the minimum. A "pattern move" is then made in this direction. Fletcher makes a comparison of a number of methods, including the search technique proposed by Swann.

Some random search techniques are discussed by Brooks (4), Favreau (10) and (11), Mitchell (28), and Munson (29). Brooks gives a general discussion of the random method and defines three types of random search: (1) simple random methods, (2) stratified random method, and (3) creeping random method. Favreau and Munson present techniques for implementing random search on the analog computer. A hybrid computer technique is presented by Mitchell which employs digital logic to implement different random search strategies and step-size changes. Random search methods are preferred for poorly behaved functions containing discontinuities or nonlinearities. These are the less elegant and optimal of all the techniques, however, they are reliable and practical if high speed computers are available. The modern repetitive digital and hybrid computers offer very attractive implementation possibilities for these techniques.

Contour mapping appears to be very popular as a desirable tool, however, automatic techniques for achieving these maps appear to be lacking (at least, in the literature). McCue (25), (26) and (27) has developed such a technique for two parameter optimization problems. Since a similar computer routine has been included in the composite program, a complete description is provided in the sequel. This is a very useful tool when difficulties are encountered in computing the roots. It may be employed to determine the number of roots, their location, division between closely spaced roots, and certain features of the function such as valleys, ridges and saddle points, which would hamper convergence.

The final conclusion obtained from the review as well as work performed in this study is that further development is required in the area of techniques for determining certain features of the equations to be solved. The justification for this conclusion is based on the following requirements:

1. The selection of a particular method for a particular function is frequently essential.
2. An initial guess for the solution is essential for all methods.

Without these additional techniques it can be extremely difficult, if not impossible in some instances, to successfully apply any of the methods.

2.2 REVIEW OF SOME PARTICULAR METHODS

The following subsections discuss in some detail those methods which were most thoroughly examined during the study contract. For the most part, these include the methods employed in the composite computer program: Newton-Raphson, Fletcher-Powell, Simplex Method, and Contour Mapping. The other methods discussed in detail are the method of False Position and the Sequential Minimax Search.

The principal reason for including the method of False Position (known as Regula-Falsi) is that it is one of the oldest and best known methods. A

computer program, user's manual, and program's listing has also been included (see Appendixes B-2 and C-2), however, it was not included in the composite program. The reasons for excluding it from the composite program were (1) its similarity to the Newton-Raphson method, and (2) the performance of the computer program was, in general, not nearly as good as that of the Newton-Raphson computer program.

The Newton-Raphson and Fletcher-Powell methods are both gradient techniques, however, the Fletcher-Powell method differs in that it is a minimization procedure and solves for the roots by forming a single function from the system of equations.

The sequential minimax search was considered in detail because it does not require an initial estimate sufficiently close to the root in question, while this is an essential requirement of all the other methods. The present formulation of the method, however, is limited to unimodal functions with one independent variable. Generalization of this method appears to be quite complicated and the effort required to develop such a generalization would be extensive. Therefore, a computer program was not attempted.

2.2.1 Newton-Raphson

Suppose that α is the desired root of the equation $f(x) = 0$; let x_1 be an abscissa near enough to α that the tangent at $P[x_1, f(x_1)]$ cuts the axis nearer to α than x_1 . This point of intersection (Figure 1) is the second approximation of x_2 .

Since the tangent at P is

$$y - f(x_1) = f'(x_1) [x - x_1]$$

it is easy to find that

$$x_2 = x_1 - f(x_1)/f'(x_1) .$$

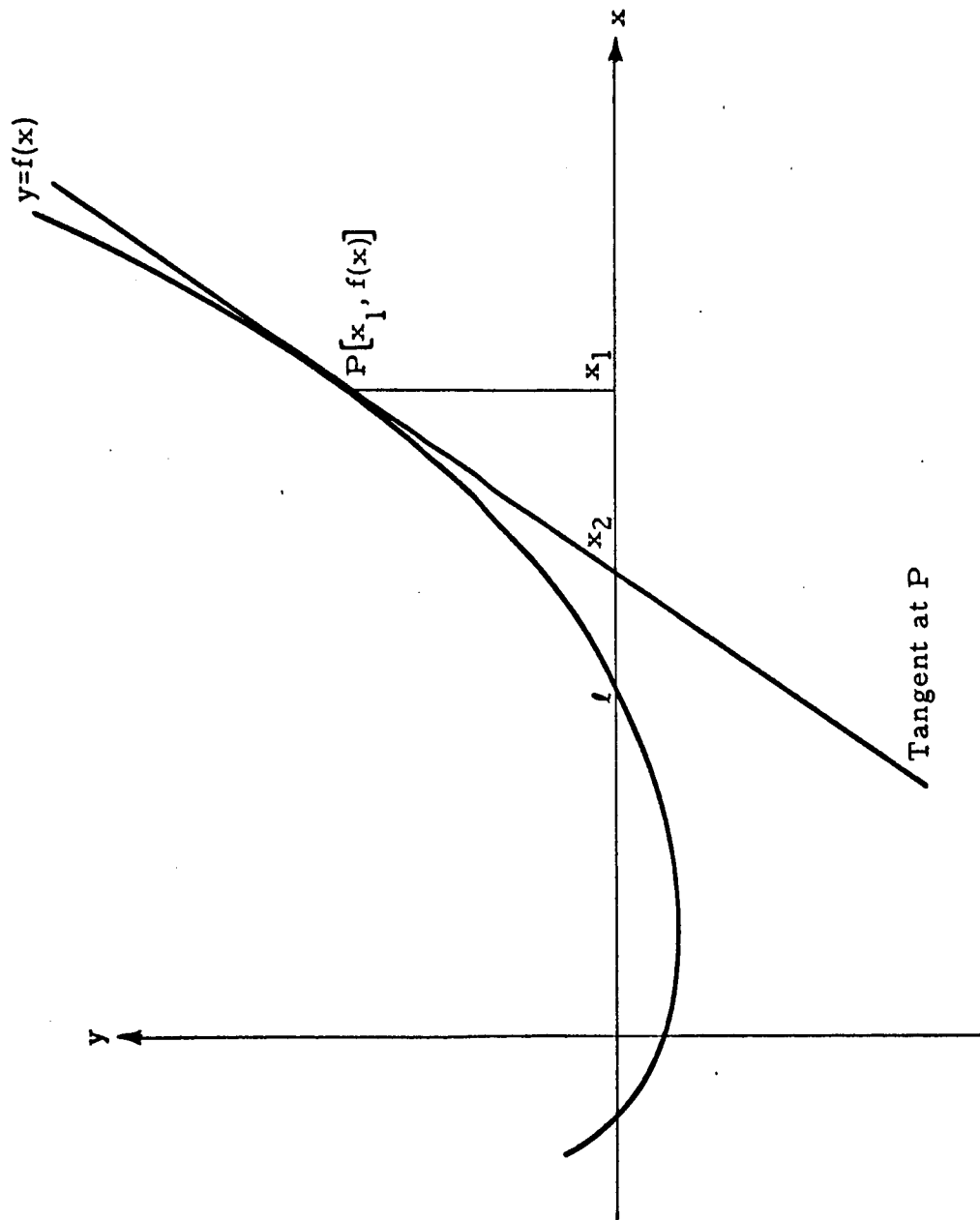


Figure 1 - Newton-Raphson Method

Using x_2 as a starting point, the tangent at $[x_2, f(x_2)]$ will give

$$x_3 = x_2 - f(x_2)/f'(x_2)$$

The process can be repeated, and the root α is approached with great speed. It is often very convenient to use the rule-of-thumb that if the correction term $f(x_i)/f'(x_i)$ begins with n zeros after the decimal point, then the result is correct to $2n$ decimals, i.e., the number of correct decimals roughly doubles at each stage.

Another worthwhile feature of the Newton-Raphson process is the fact that it is self-correcting for minor errors. Any errors made in determining x_2 will merely give a different point from which to draw the second tangent; this will not affect the limit α approached by the sequence x_1, x_2, x_3, \dots .

Newton-Raphson is easily extended to simultaneous solution of systems of equations. Considering the system

$$\begin{cases} f_1(x_1, \dots, x_n) = 0 \\ \vdots \\ f_n(x_1, \dots, x_n) = 0 \end{cases} \quad (1)$$

where $n \geq 20$

we obtain corrections $\Delta x_1, \dots, \Delta x_n$ for the estimates to the roots x_1, \dots, x_n so that new estimates may be obtained by

$$\begin{cases} x_{1(i)} = x_{1(i-1)} + \Delta x_{1(i)} \\ x_{2(i)} = x_{2(i-1)} + \Delta x_{2(i)} \\ \vdots \\ x_{n(i)} = x_{n(i-1)} + \Delta x_{n(i)} \end{cases} \quad (2)$$

where

$$\begin{cases} f_1(x_{1(i)}, x_{2(i)}, \dots, x_{n(i)}) = 0 \\ \vdots \\ f_n(x_{1(i)}, x_{2(i)}, \dots, x_{n(i)}) = 0 \end{cases} \quad (3)$$

By expanding (3) by Taylor's theorem for a function of n-variables we get at the i^{th} iteration

$$\begin{cases} F_{x_1}^1 + F_{x_2}^1 + \dots + F_{x_n}^1 = -f_1(x_1, x_2, \dots, x_n) \\ F_{x_1}^2 + F_{x_2}^2 + \dots + F_{x_n}^2 = -f_2(x_1, x_2, \dots, x_n) \\ \vdots \\ F_{x_1}^n + F_{x_2}^n + \dots + F_{x_n}^n = -f_n(x_1, x_2, \dots, x_n) \end{cases}$$

where

$$F_{x_1}^1 = \left. \frac{\partial f_1}{\partial x_1} \right|_{(i-1)} \Delta x_1, \quad F_{x_1}^2 = \left. \frac{\partial f_2}{\partial x_1} \right|_{(i-1)} \Delta x_1, \text{ etc.}$$

We now have the matrix form $AX = B$ where

$$A = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \dots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \dots & \frac{\partial f_n}{\partial x_n} \end{bmatrix}$$

$$x = \begin{bmatrix} \Delta x_1 \\ \Delta x_2 \\ \vdots \\ \Delta x_n \end{bmatrix}$$

$$B = \begin{bmatrix} -f_1(x_1, x_2, \dots, x_n) \\ -f_2(x_1, x_2, \dots, x_n) \\ \vdots \\ -f_n(x_1, x_2, \dots, x_n) \end{bmatrix}$$

We then solve for x using the Gauss-Jordan reduction technique thereby obtaining Δx_j ($j=1, \dots, n$). We then reiterate until all $\Delta x_j < \epsilon$ for some prescribed accuracy ϵ .

2.2.2 False Position

One of the oldest methods of finding roots is known as Regula-Falsi (method of False Position). It requires a knowledge of the approximate location of the root and the computation of two values $f(a)$ and $f(b)$, where $a < r < b$, r being a root of $f(x) = 0$ (Figure 2). If a and b are close enough to r so that no other root lies between a and b , then by continuity $f(a)$ and $f(b)$ are of opposite sign. If we replace the arc AB by the chord AB , we obtain an abscissa c which is closer to r than A was (in Figure 2, $f(c)$ is negative). The value of c is obviously $[af(b) - bf(a)]/[f(b) - f(a)]$. The process may be repeated using the chord BC . Hanry and Bernede (Reference 3) have developed a computer scheme using Regula-Falsi which is capable of solution of ten equation systems, providing transcendental terms are not numerous. Accuracy of solution and computer execution time are not acceptable above the five-equation system.

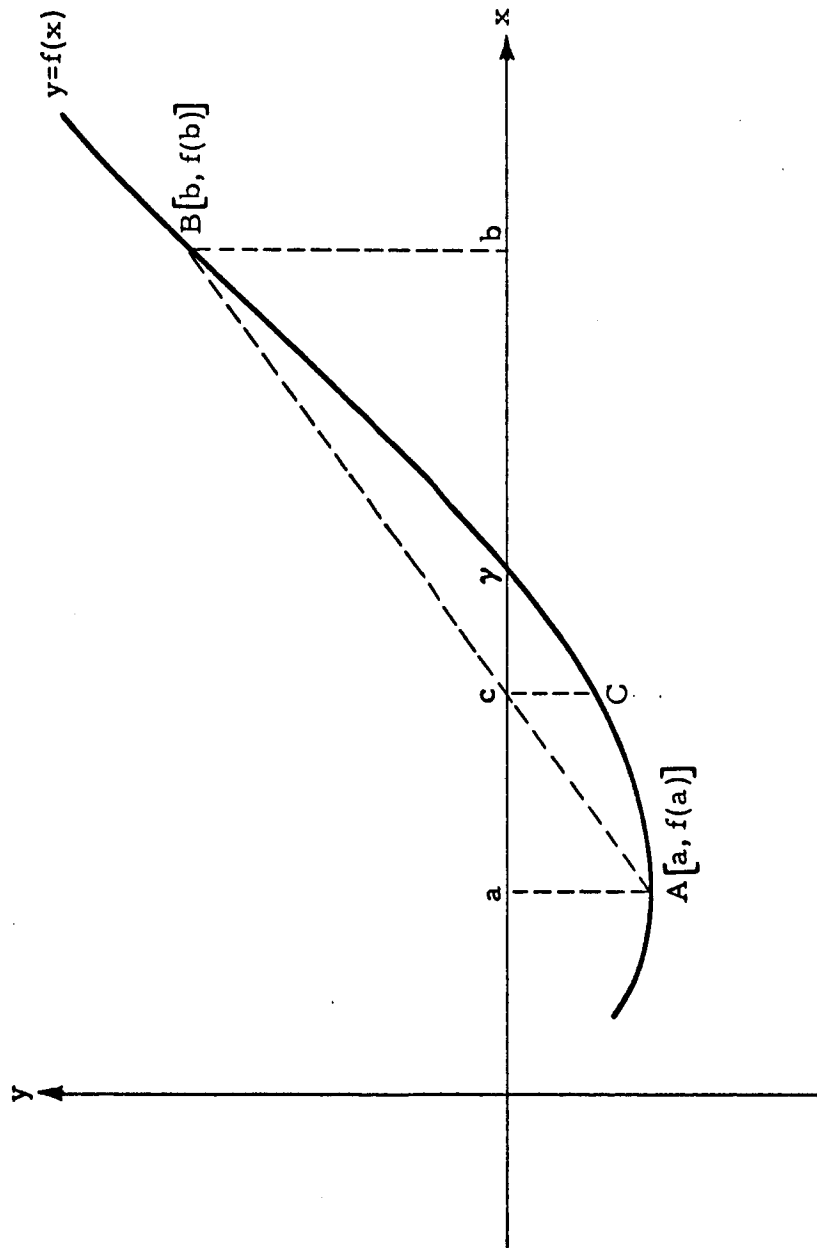


Figure 2 - Method of False Position

2.2.3 Fletcher-Powell Method

Minimization procedures may be employed to solve systems of non-linear equations. Given the system

$$\begin{aligned} f_1(\bar{x}) &= 0 \\ &\cdot \\ &\cdot \\ &\cdot \\ f_n(\bar{x}) &= 0 \end{aligned} \quad (4)$$

where \bar{x} is the vector with components x_1, \dots, x_n , the function $f(\bar{x})$ is formed

$$f(\bar{x}) = f_1^2(\bar{x}) + \dots + f_n^2(\bar{x}) \quad (5)$$

The function in (5) is non-negative and achieves the minimum value zero only when the system (4) is satisfied. The value \bar{x} which minimizes (5) therefore satisfies (4) (i.e., it is a root of the system). Hence (4) may be solved by locating the minimum of (5).

The theory of this method has been developed for quadratic functions of n variables. It is known that even if the function to be minimized is non-quadratic, the second-order terms of the Taylor series expansion dominate in the vicinity of the minimum. Therefore the only methods which will converge quickly for a general function are those which will guarantee to find the minimum of a general quadratic speedily.

Let it be required to minimize a quadratic function of the n variables x_1, x_2, \dots, x_n . Denote the column vector $(x_1, \dots, x_n)^T$ by \bar{x} . The quadratic form to be minimized may be written

$$\begin{aligned} f(\bar{x}) &= b + \sum_1^n a_i x_i + \frac{1}{2} \sum_{i,j} G_{ij} x_i x_j \\ &= b + \bar{a}^T \bar{x} + \frac{1}{2} \bar{x}^T G \bar{x} \end{aligned} \quad (6)$$

In this representation, b is a scalar constant, \bar{a}^T is the row vector (a_1, \dots, a_n) and G is the non-singular, symmetric matrix with elements G_{ij} . The gradient of the function f at the point \bar{x} is

$$\overline{g(\bar{x})} = \bar{a} + G \bar{x} \quad (7)$$

At the minimum point \bar{x}_m the gradient vanishes, so that

$$0 = \bar{a} + G \bar{x}_m \quad (8)$$

Subtracting (8) from (7) and pre-multiplying by G^{-1} one finds

$$\bar{x} - \bar{x}_m = G^{-1} \overline{g(\bar{x})} \quad (9)$$

as the displacement between the point \bar{x} and the minimum point \bar{x}_m . Clearly, if one were interested in the minimization of quadratic functions only, one would simply compute \bar{x}_m directly from (9). Inasmuch as this of course would not provide the correct answer when the given function is non-quadratic, the following iterative procedure, in which G^{-1} is not evaluated directly, is employed for general functions. If the function happens to be quadratic (in n variables) the procedure converges to the minimum in precisely n steps, and the method in this case requires about the same amount of computer time as the direct use of Equation (9). For non-quadratic functions, more than n steps will be required. The method is as follows:

1. Starting with an initial estimate \bar{x}^0 of the minimum compute the function $f(\bar{x}^0)$ and the gradient $\overline{g(\bar{x}^0)} \equiv \bar{g}^0$ for brevity.

2. Set $\bar{s}^0 = -H^0 \bar{g}^0$

The matrix H^0 is any positive definite symmetric matrix (of dimension $n \times n$). It is convenient to let $H^0 = I$. This will cause the initial direction in the descent process (Step 3) to be along the line of steepest descent.

3. Move from the point \bar{x}^0 along the line $\bar{x}^0 + \lambda \bar{s}^0$ until $f(\bar{x}^0 + \lambda \bar{s}^0)$ is a minimum with respect to λ . Let the critical value of λ be α^0 , which can be shown to be positive.

4. Set $\bar{\sigma}^0 = \alpha^0 \bar{s}^0$

5. Set $\bar{x}^1 = \bar{x}^0 + \bar{\sigma}^0$

6. Compute $f(\bar{x}^1)$ and $g(\bar{x}^1)$

7. Set $\bar{y}^0 = \bar{g}^1 - \bar{g}^0$

8. Define $A^1 = \frac{\bar{\sigma}^0 \bar{\sigma}^{0T}}{\bar{\sigma}^{0T} \bar{y}^0}$, $B^1 = \frac{-H^0 \bar{y}^0 \bar{y}^{0T} H^0}{\bar{y}^{0T} H^0 \bar{y}^0}$

and set $H^1 = A^1 + B^1$

9. Repeat the entire process proceeding from the point \bar{x}^1 , with the gradient \bar{g}^1 and matrix H^1 . Continue in this way.

The predicted absolute distance from the minimum is (from Eq. 9)

$$d^i = \left[\bar{g}^i T G^{-1} G^{-1} \bar{g}^i \right]^{1/2}$$

$$\approx (\bar{s}^i T \bar{s}^i)^{1/2}$$

The last line of this equation comes from the definition $\bar{s}^i = -H^i \bar{g}^i$ (Step 2 of the procedure) and the fact that H^i tends to G^{-1} . The procedure just outlined may be terminated when the distance d^i is less than some prescribed amount, or alternatively when every component of \bar{s}^i is less than a prescribed accuracy. Additional safeguards stated in Reference 4 are to work through at least n iterations and to apply the tests to $\bar{\sigma}^i$ as well as \bar{s}^i .

The following points with respect to the Fletcher-Powell Method are proved in Reference 4:

1. The method is stable, in that the function to be minimized is decreased by each step.
2. For quadratic functions, the minimum is found in n iterations, and $H^n = G^{-1}$.

The procedure used for implementing Step 3, that is the determination of the minimum of f along the line $\bar{x}^i + \lambda \bar{s}^i$ requires the calculation of the function and the gradient at the point \bar{x}^i and a point $\bar{x}^i + \lambda \bar{s}^i$ on this line. Cubic interpolation is used to locate the minimum of the function along the line.

2.2.4 Simplex Search Method

Various search routines for locating the minimum (or maximum) of a function have been devised. These are based upon the principle of evaluating the function at points selected according to a certain strategy. Under the proper circumstances these procedures converge to the minimum or maximum in question.

The celebrated search method for functions of a single variable is that due to J. Kiefer (Reference 5). This method requires that the function be unimodal; i.e., that there be a single maximum and that the function be strictly increasing to the left and strictly decreasing to the right of the maximum. For a specified number of function evaluations to be made, the procedure provides an interval of smallest length (in the ϵ -minimax sense defined in Reference 5) containing the maximum, the points being selected according to the Fibonacci sequence.

For functions of several variables, D. J. Newman (Reference 6) gives a procedure for locating the maximum based upon evaluating the function at a minimum number of points. If k is the number of independent variables, a unit cube containing the maximum is selected in k -space. The method determines a point for which the value of the function dominates all function values on the $(n + 1)^k$ lattice points, n being arbitrary. Unimodality is postulated and the author is explicit for the case of two variables. Hooke and Jeeves in Reference 7 describe direct search and pattern search techniques. An elegant simplex method described by Nelder and Mead (Reference 8), having the advantage of a simple logic, has been included in the composite program.

The simplex method is diagrammed in the flow chart in Appendix C-1 for a function of an arbitrary number of variables. The method may be described as follows. One starts by selecting the vertices of a simplex (see Reference 9 for example for a clear definition) in the k -dimensional space of the function to be minimized. In two variables this means selecting the three vertices of a non-degenerate triangle; in three variables it means picking the vertices of a non-degenerate tetrahedron, etc. The simplex should preferably be located not too far from the position of the minimum which is being sought. Denote the $k+1$ points by P_0, \dots, P_k :

Compute $y_0 = f(P_0), \dots, y_k = f(P_k)$ where f is the function to be minimized. Denote the maximum of these numbers by $y_h = f(P_h)$ and the minimum by $y_l = f(P_l)$. Compute the centroid of the set of all the $k+1$ points excluding the highest, P_h :

$$\bar{P} = \frac{1}{k} \sum_{i \neq h} P_i$$

Replace P_h by P^* defined by

$$P^* = (1 + \alpha) \bar{P} - P_h \quad (\alpha = \text{reflection coefficient, } \alpha > 0)$$

where P^* is on the opposite side of P from P_h and on the line joining them. Compute $y^* = f(P^*)$.

If $y^* < y_l$ a new minimum has been determined and therefore one decides to move further in the same direction by expanding to a new point P^{**}

$$P^{**} = (1+\gamma) P^* - \gamma \bar{P}$$

where $\gamma > 0$ and $1 + \gamma =$ expansion coefficient. Compute $y^{**} = f(P^{**})$. If $y^{**} < y_l$ replace y_h by y^{**} (all other k points of the original simplex remain unaltered). Restart the process, again labeling the high point and the low point of this new simplex y_h and y_l . If $y^{**} > y_l$ the expansion has failed. In that case, remove the point P^{**} , replace P_h by P^* , and restart the process.

However, if the point P^* determined by the reflection process was such that $y^* > y_i$ for all $i \neq h$, then a new P_h should be defined as either the old P_h or P^* (whichever has the lower y value) and form

$$P^{**} = \beta P_h + (1 - \beta) \bar{P}$$

This is a contraction, the point P^{**} being closer to \bar{P} than P_h . The quantity β is known as the contraction coefficient, and $0 < \beta < 1$. Compute $y^{**} = y(P^{**})$. If $y^{**} \leq y_h$ then P_h is replaced by P^{**} and the process is restarted. However, if $y^{**} > y_h$ the contraction has failed, in which case all P_i 's are replaced by $\frac{1}{2}(P_i + P_l)$ and the process is restarted. The iteration continues until the minimum is reached.

Figure 3 illustrates this method for the solution of the system of equations

$$x^2 + y - 11 = 0$$

$$x + y^2 - 7 = 0$$

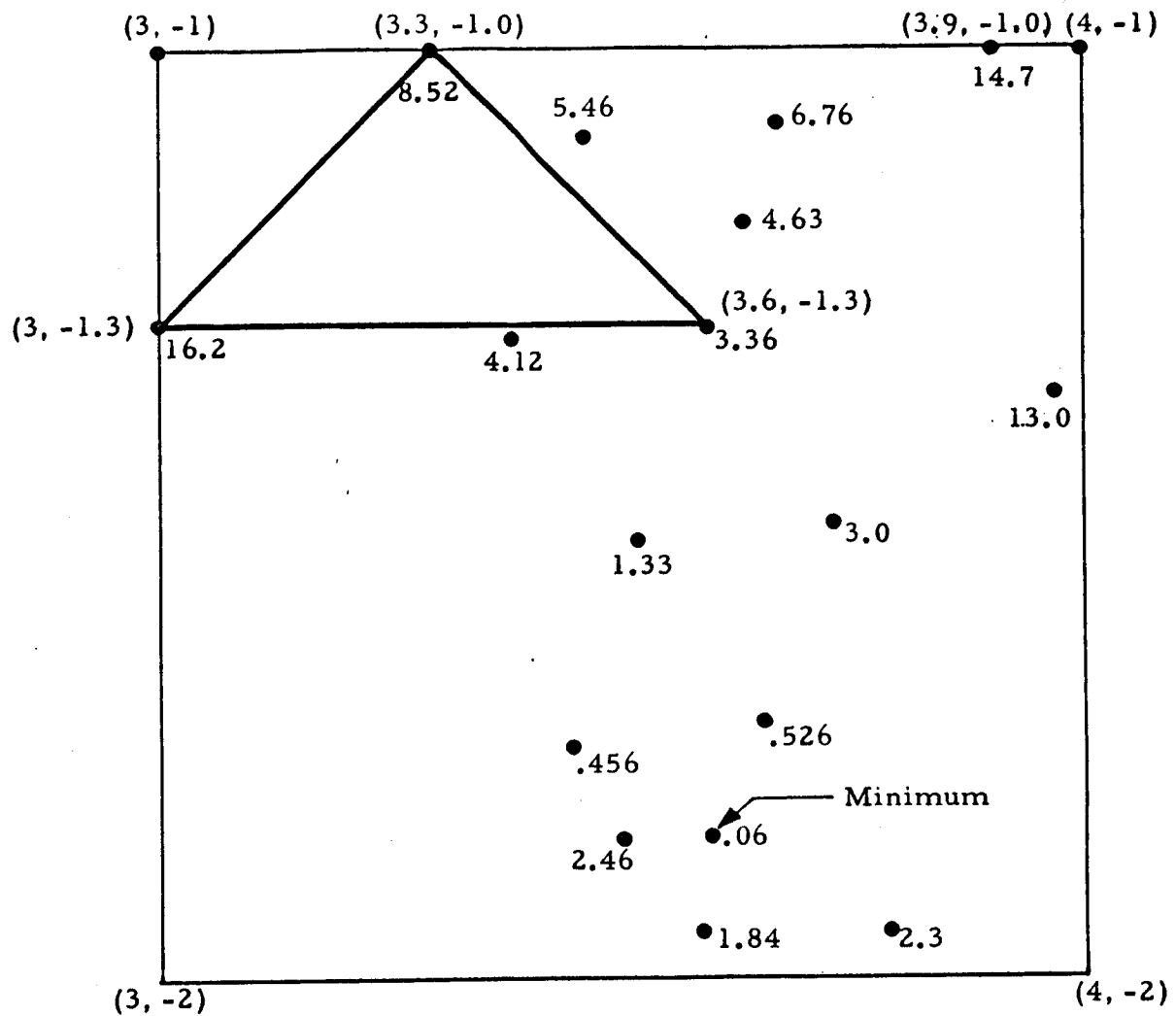


Figure 3 - Solution of the System of Equations $x^2 + y - 11 = 0$,
 $x + y^2 - 7 = 0$

which is accomplished by minimizing the function

$$f(x, y) = (x^2 + y - 11)^2 + (x + y^2 - 7)^2$$

The three initial simplex vertices are selected as (3, -1.3), (3.3, -1.0), (3.6, -1.3). After 39 iterations the procedure gives (3.5844284, -1.8481265) for the minimum point. The figure shows the initial simplex and points that were determined as a result of the reflection, expansion, and contraction steps that take place as a result of the method described. The value of the function is indicated next to each point.

To ensure that the simplex method will not seek an incorrect minimum point, a unit cube in k-space should be determined which contains the desired minimum and the function should be redefined with some arbitrarily high value on the outside of the cube. Any attempt to wander outside this region will cause the process to return to the inside of the cube and converge to the desired minimum.

The simplex method was employed successfully on another problem

$$(x_1 + 10x_2)^2 + 5(x_3 - x_4)^2 + (x_2 - 2x_3)^4 + 10(x_1 - x_4)^4 = \min.$$

with the initial simplex vertices at the five points

$$(3, -1, 0, 1)$$

$$(2, -1, 0, 1)$$

2.2.5 Sequential Minimax Search

The various iterative approaches for solving systems of equations, such as Newton-Raphson and Fletcher-Powell, require an initial estimate sufficiently close to the root in question.

Relatively recently some interest has been displayed in obtaining search procedures for locating approximately the maximum (or minimum) of a function of one or more variables by computing the function at a minimum number of points. Kiefer in Reference 5 develops an optimum sequential search strategy for locating the maximum of a unimodal function of one variable. This is a function $f(x)$ defined in the unit interval for which there exists a number x_m in the interval such that either

- a. $f(x)$ is strictly increasing for $x \leq x_m$ and strictly decreasing for $x > x_m$, or
- b. $f(x)$ is strictly increasing for $x < x_m$ and strictly decreasing for $x \geq x_m$.

Unimodality is the only assumption. Assumptions with regard to continuity, differentiability, etc., are not required. Let \mathcal{S}_N denote the class of all strategies involving N successive observations (or evaluations) of the function, and let $D(f, S)$ denote the interval containing the maximum point obtained as a result of N observations of the function f by means of the strategy S . Also let $L(D)$ denote the length of D . The problem is to find a strategy $S_N^* \in \mathcal{S}_N$ such that, for $\epsilon > 0$

$$\sup_f L[D(f, S_N^*)] \leq \inf_{S \in \mathcal{S}_N} \sup_f L[D(f, S)] + \epsilon \quad (10)$$

Let U_n be the n^{th} Fibonacci number (i.e., $U_n = U_{n-1} + U_{n-2}$). Thus

$$\begin{array}{ll} U_0 = 0 & U_4 = 3 \\ U_1 = 1 & U_5 = 5 \\ U_2 = 1 & U_6 = 8 \\ U_3 = 2 & U_7 = 13 \end{array}$$

S_2^* is defined by the observations at the two points

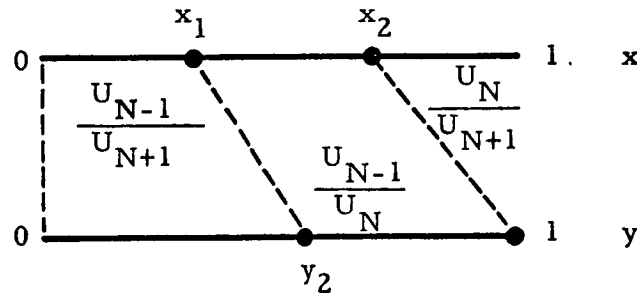
$$x_1 = \frac{1}{2}, x_2 = \frac{1}{2} + \epsilon$$

and

$$D(f, S_2^*) = \begin{cases} [0, x_2] & \text{if } f(x_1) \geq f(x_2) \\ [x_1, 1] & \text{if } f(x_1) < f(x_2) \end{cases}$$

The strategy S_N^* is defined inductively as follows: Suppose S_{N-1}^* has been defined for $N \geq 3$. S_N^* is obtained by choosing the points $x_1 = \frac{U_{N-1}}{U_{N+1}}$, $x_2 = 1 - x_1 = \frac{U_N}{U_{N+1}}$. Suppose $f(x_1) \geq f(x_2)$. Then make the change of variable

$$y = h(x) = x \frac{U_{N+1}}{U_N}.$$



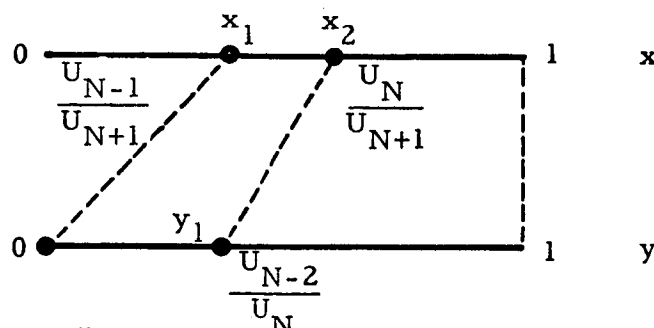
The point $x = x_2$ is mapped to the point $y = 1$, $x = 0$ is mapped into $y = 0$,

and $x = x_1$ is mapped to the point $y = y_2 = \frac{U_{N-1}}{U_N}$. Since the assumption

$f(x_1) \geq f(x_2)$ it follows that the maximum point is in the interval $[0, x_2]$ in the x domain, or in $[0, 1]$ in the y domain. Write $f^*(y) = f(x)$ and start all over again in the y domain, this time allowing y_2 to play the role that x_2 did previously. This may be accomplished by replacing N by $N-1$, and then using the strategy S_{N-1}^* on the function $f^*(y)$. Note that the observation $f^*(y_2) = f(x_1)$ has already been made and the next observation is made at $y_1 = 1 - y_2 = \frac{U_{N-2}}{U_N}$. Thus, y_1 has the role of x_1 with N changed to $N-1$.

Suppose instead that $f(x_1) < f(x_2)$, which implies that the maximum point is in $[x_1, 1]$. In this case the change of variable $y = h(x) = \frac{-U_{N-1} + xU_{N+1}}{U_N}$

is made. Note now that x_1 is mapped into zero in the y domain, the point $x = 1$ is mapped into $y = 1$, and x_2 is mapped into $\frac{U_{N-2}}{U_N}$. Call this point y_1 . Note that y_1 plays the role of x_1 in the sense that N is replaced by $N-1$ and it is now possible to use S_{N-1}^* on the function $f^*(y)$ in the



unit y interval. Since S_{N-1}^* has been defined (induction hypothesis) the description of the strategy S_N^* is complete. Under this strategy the length of the interval obtained after N observations containing the maximum point satisfies

$$L \left[D(f, S_N^*) \right] \leq \epsilon + \frac{1}{U_{N+1}}. \quad (11)$$

Kiefer in Reference 5 proves that the strategy S_N^* just described fulfills the condition (10).

The solution of systems of equations in several unknowns requires the minimization of a function of several variables. D. J. Newman in Reference 6 attempts to generalize Kiefer's procedure to higher dimensions. Letting k denote the number of variables (or dimensions), Newman seeks the maximum value of the unimodal function in the unit cube in k -space, but restricts consideration to the function as defined at the $(k+1)^n$ lattice point (n arbitrary).

The problem then is to determine a point, not necessarily a lattice point, which dominates all of the values of the function on the lattice points. In two variables ($k = 2$) an explicit procedure is given and it is shown that the number of observations $C(2, n)$ satisfies

$$C(2, n) \leq 90 \log (n+1).$$

The procedure is quite complicated, but if an additional restriction is made, viz that the function be C^1 -unimodal, then a practical procedure exists such that the number of observations required is

$$C^1(2, n) < 10 \log (n+1) + 6.$$

In general, for k dimensions it is stated that the number of function evaluations $C(k, n)$ required is

$$C(k, n) \leq C_k \log n.$$

However, no formula for C_k is given in the article for $k > 2$.

2.2.6 Contour Plot

The contour plotting routine is an aid in locating the initial estimates to the roots of a system. The number of contours desired along with the grid limits are input by the user. A plane must be selected for all systems greater than two. The technique employed here is a simplification to that in Reference 10.

The technique divides the grid into n squares across and n squares down. The number pairs (x, y) , where each grid line crosses, are determined and the value of the system at each intersection is computed. The largest functional value is F_{\max} and the smallest is F_{\min} . The user's choice of a scale factor, SF , decides the spacing of the contours between these limits.

If equal increments in F_c are required, then a scale factor of unity is input and we compute

$$F_{c_i} = F_{\min} + (i + 1) \Delta F$$

where

$$\Delta F = (F_{\max} - F_{\min}) / (N - 1).$$

A scale factor greater and less than unity will group the contours closest together around a zero and pole respectively. In this case we use

$$F_{c_1} = F_{c_{\min}} ; F_{c_i} = \Delta F_{\min} + (SF)^{i-1} \Delta F \quad (i = 2, \dots, N)$$

where

$$\Delta F = (F_{\max} - F_{\min}) / (SF)^{N-1}$$

where N is the number of contour plots desired.

The routine numbers each grid line x and y intersection as follows:

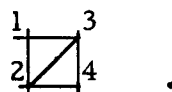
1	3	5	7			67	69	71
2	4	6	8			68	70	72
73	75							143
74	76							144
...
...	1293		1295
...	1294		1296

Considering the first box, the McCue's routine (Reference 10) determines the diagonal representing the vertices with the least difference:

If $f(x_1, y_1) - f(x_4, y_4) \leq f(x_3, y_3) - f(x_2, y_2)$ then

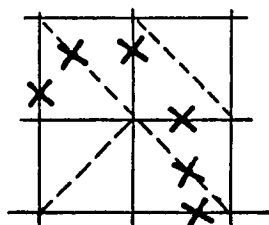


but if $f(x_1, y_1) - f(x_4, y_4) > f(x_3, y_3) - f(x_2, y_2)$ then



This is done for each box.

Then the sides and diagonals are surveyed to determine the location of the contour value F_c . If $F_1 > F_c > F_2$ then a linear interpolation is performed to determine the (x, y) values where F_c crosses the side (1, 2) of box 1. Then sides (2, 4) and the diagonal are tested. Finally, the other half of the box is tested and so forth through all boxes of the grid frame. The interpolated points are plotted, such as



By connecting the points, a contour plot is obtained.

In the simplified version, a continuous line is not plotted; only points along the contour are plotted. This eliminates the principal advantage gained by including the interpolation on the diagonal. Interpolation on the diagonal provides directional information for plotting a line contour. Therefore, the diagonal is omitted from the routine included in the present program. Also, the present program employs a grid of 35 squares down and 35 squares across.

It should be pointed out that a tighter grid will result in points so closely spaced that the result would appear as a continuous line. This might be a better approach than including the diagonal and connecting points with straight line segments. For clarity, the printed grid does not have to be nearly as tight as that employed in the computation. These comments are intended as recommendations for additional minor program modifications.

An example of a contour plot obtained from the present routine, described in Appendix C-1, is shown in Figures 4a through 4e. This example is the test problem described in Section 2.4.1 for closely spaced roots with $\epsilon = 0.005$. The point symbols correspond to F_c contour values as follows:

Fig. No.		4a	4b	4c	4d	4e
F_{\min}		.00319	.00319	.00319	.00319	.00319
F_{\max}		33.95	33.95	33.95	33.95	33.95
S.F.		1.0	1.4	1.6	1.8	4.0
N		8	10	10	10	10
i = 1	O	.0032	.0032	.0032	.0032	.0032
2	O	4.85	2.30	.79	.31	.0037
3	x	9.70	3.22	1.27	.56	.0053
4	□	14.55	4.51	2.03	1.00	.012
5	Y	19.40	6.32	3.24	1.80	.036
6	+	24.25	8.84	5.18	3.24	.14
7	*	29.10	12.37	8.29	5.82	.53
8	E	33.95	17.32	13.26	10.48	2.12
9	U		24.25	21.22	18.86	8.49
10	O		33.95	33.95	33.95	33.95

493370
001 000

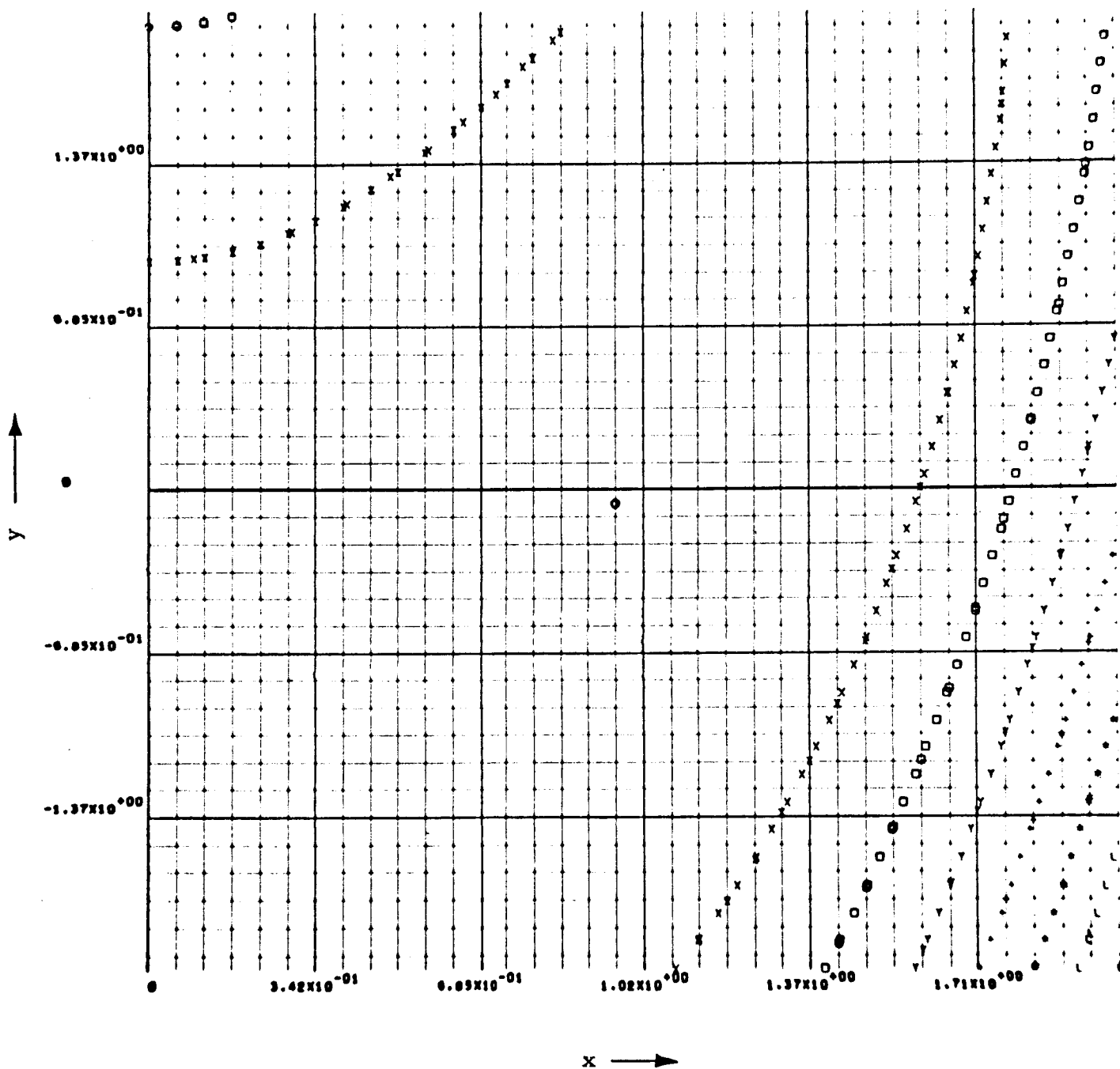


Figure 4a - Contour Plot Obtained from Computer Routine
SF = 1.0

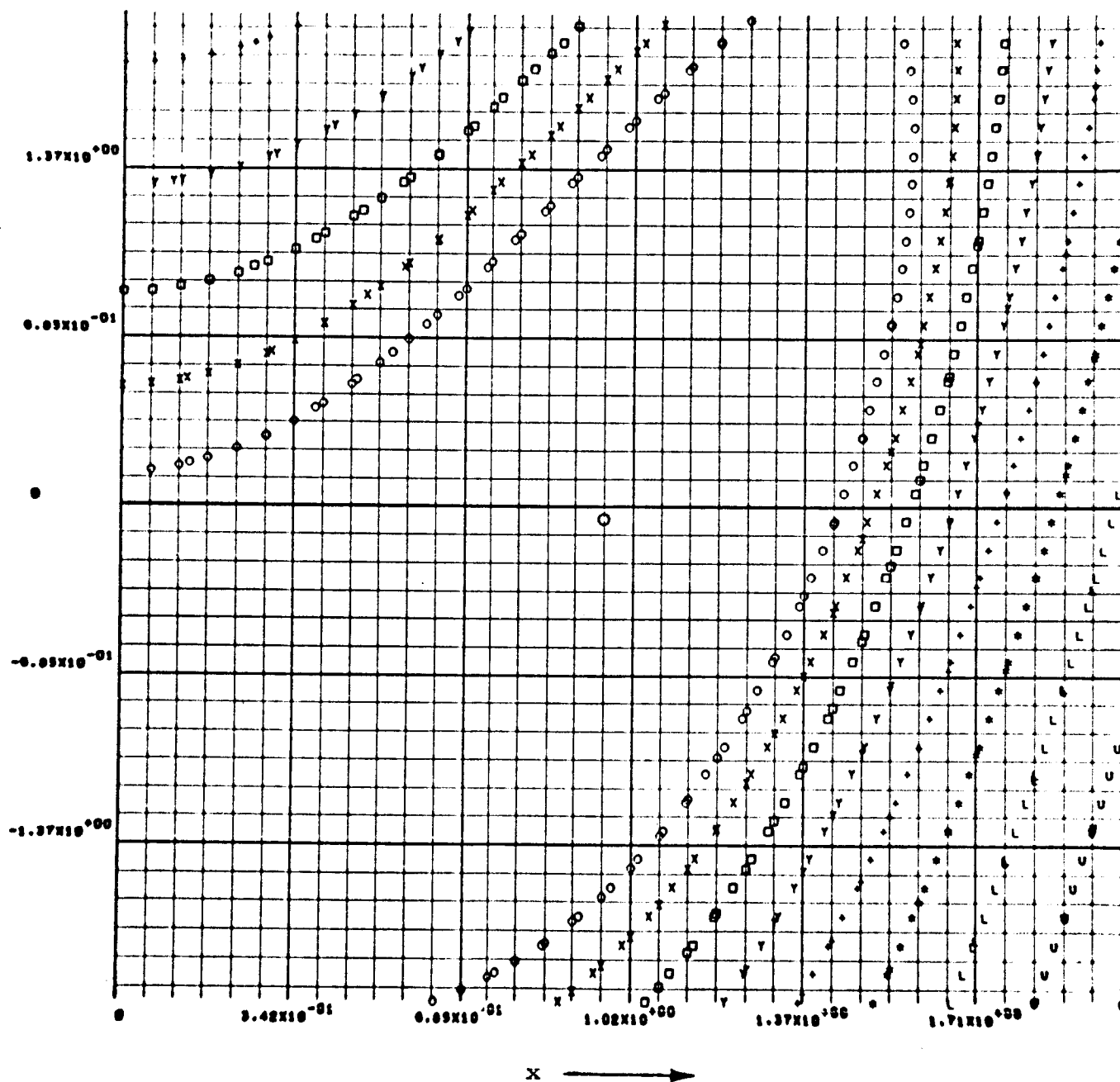


Figure 4b - Contour Plot Obtained from Computer Routine
SF = 1.4

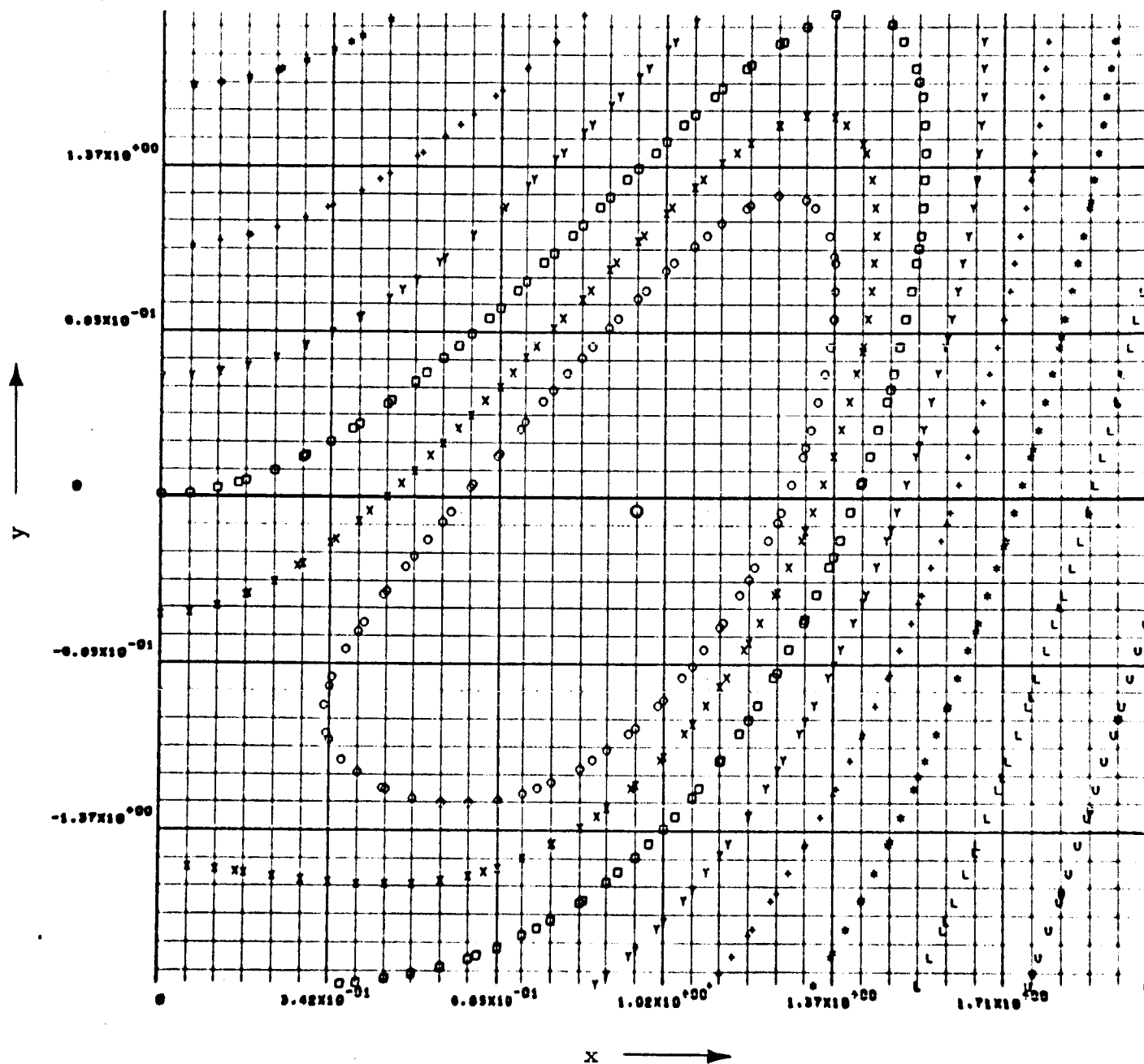


Figure 4c - Contour Plot Obtained from Computer Routine

SF = 1.6

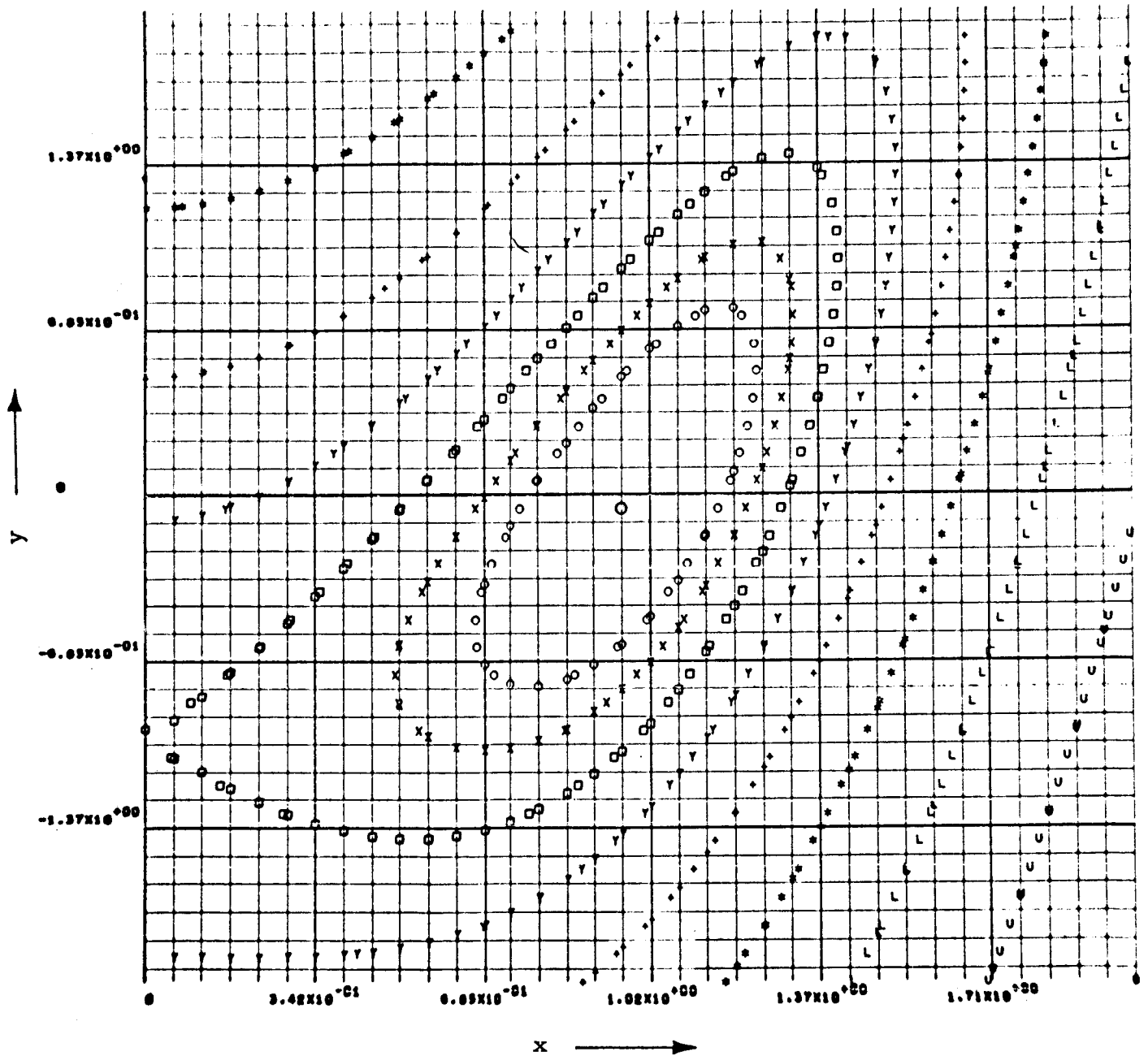


Figure 4d - Contour Plot Obtained from Computer Routine
SF = 1.8

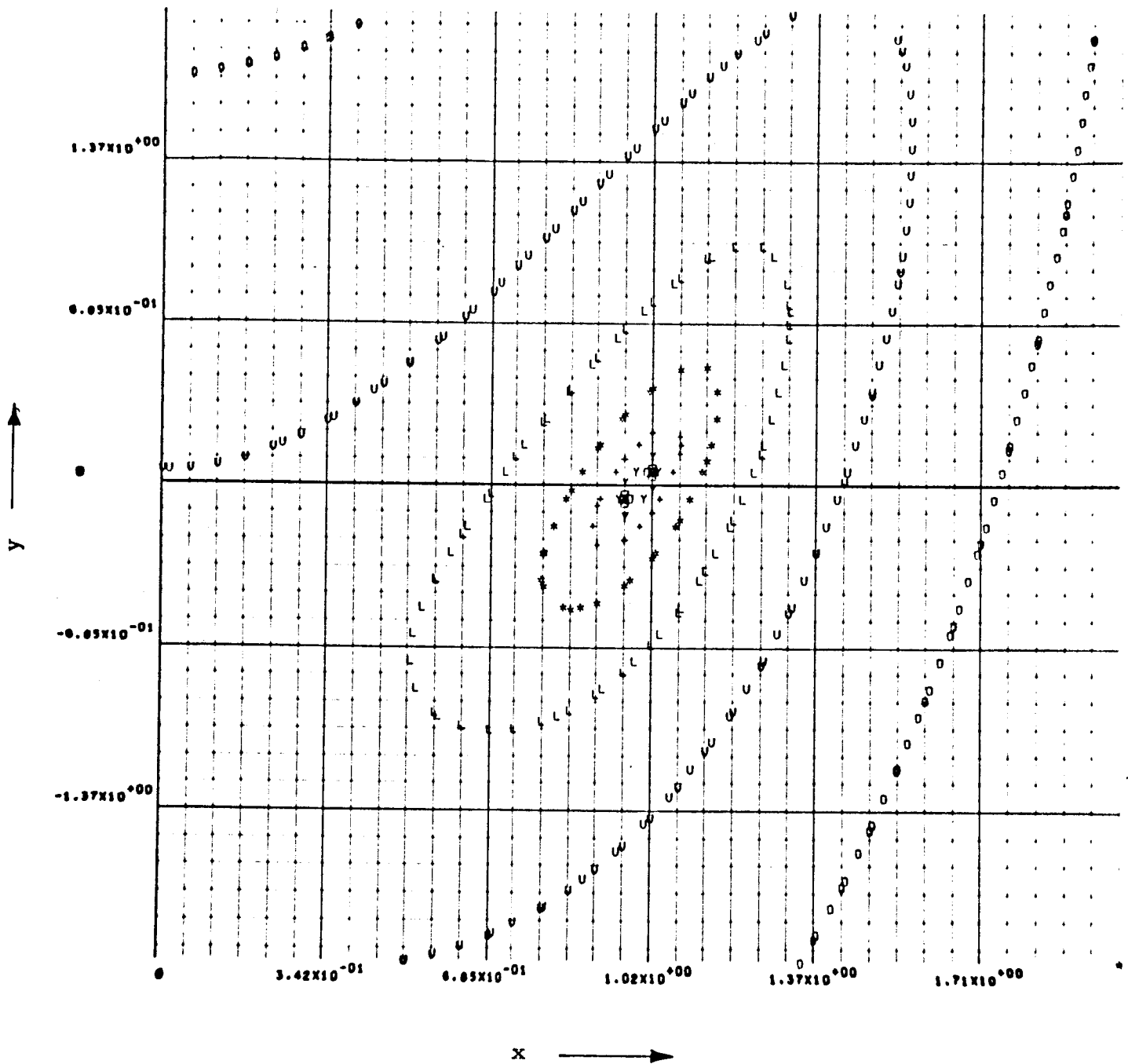


Figure 4e - Contour Plot Obtained from Computer Routine
SF = 4.0

As may be observed in Figure 4, the printed grid does not correspond to the grid used in the computation.

2.3 ANALOG AND HYBRID COMPUTATION SCHEMES

Certain aspects of the problem and the methods for seeking solutions appear to be amenable with the features for which analog and hybrid computers are attractive. Therefore, a brief study regarding such approaches and the feasibility of a hybrid computation scheme was included.

2.3.1 Analog Solutions

Analog solutions to polynomial equations may be grouped into three machine-oriented classes: (1) scanning techniques, in which the computer sweeps the complex plane in a predetermined manner and locates the points where the polynomial vanishes; (2) nulling techniques, in which the computer solves the polynomial by finding a path leading to a root such that a suitable defined error function is reduced to zero; and (3) tracking mode, in which the computer indicates the variation of a root that was previously determined by a nulling technique, if one or several coefficients of the polynomial are being changed. Examples of these methods are found in Reference 11. Problems are encountered in terms of computer elements and accuracy of solution in expanding these methods to systems of algebraic equations.

Karplus and Soroka consider many standard techniques for solving nonlinear algebraic equations on the analog computer although several are not solely for electronic analog computers (Reference 12). Potentiometric machines for real coefficients and roots are evaluated as well as the integrator solution suggested by Prof. C. P. Atkinson in which an n^{th} degree polynomial is converted into an n^{th} order ordinary differential equation and then integrated to obtain the various roots. The method of harmonic synthesis may be employed for solving high-degree algebraic equations with complex roots. An electro-mechanical harmonic synthesizer, consisting of a potentiometric equation solver with alternating voltages of adjustable phase provided to excite the system, is a possible analog technique. Electric field

representation of analytic functions applied to solutions of algebraic equations and electromagnetic field solution may be extremely successful in locating roots.

A scan method (Reference 11) for solving for the real roots of a two equation system is shown as a circuit diagram in Figure 5. Integrators are used in place of multipliers in order to avoid amplifying the noise level in the circuit. This method is suitable for finding approximations to all real roots of a system.

The accuracy of the analog computer is limited by component imperfections. A single operation can usually be performed with errors in the order of 0.01%. A typical large problem may be thought of as being accurate to about 1%, although such estimates are difficult to derive. The resulting error can be considered as a noise signal for purposes of estimating system error.

Static errors are easily assessed and are fairly well known for standard components. Dynamic errors are more difficult to assess; they arise because the analog computer elements have non-ideal dynamic response. For example, even if a summer had perfect static accuracy, errors would arise when the inputs are changing rapidly for it behaves as a low-pass network.

Johnson (Reference 13) has mentioned problems of solution stability using standard analog techniques for certain systems of equations. Gephart (Reference 14) has developed a method of setup of algebraic equations that ensures computer stability without algebraic manipulation of the system. The main disadvantage, as is true with virtually all analog techniques, is that the accuracy is determined by the complexity and size of the system of equations. The advantage of analog techniques is that no initial estimate of the root is required.

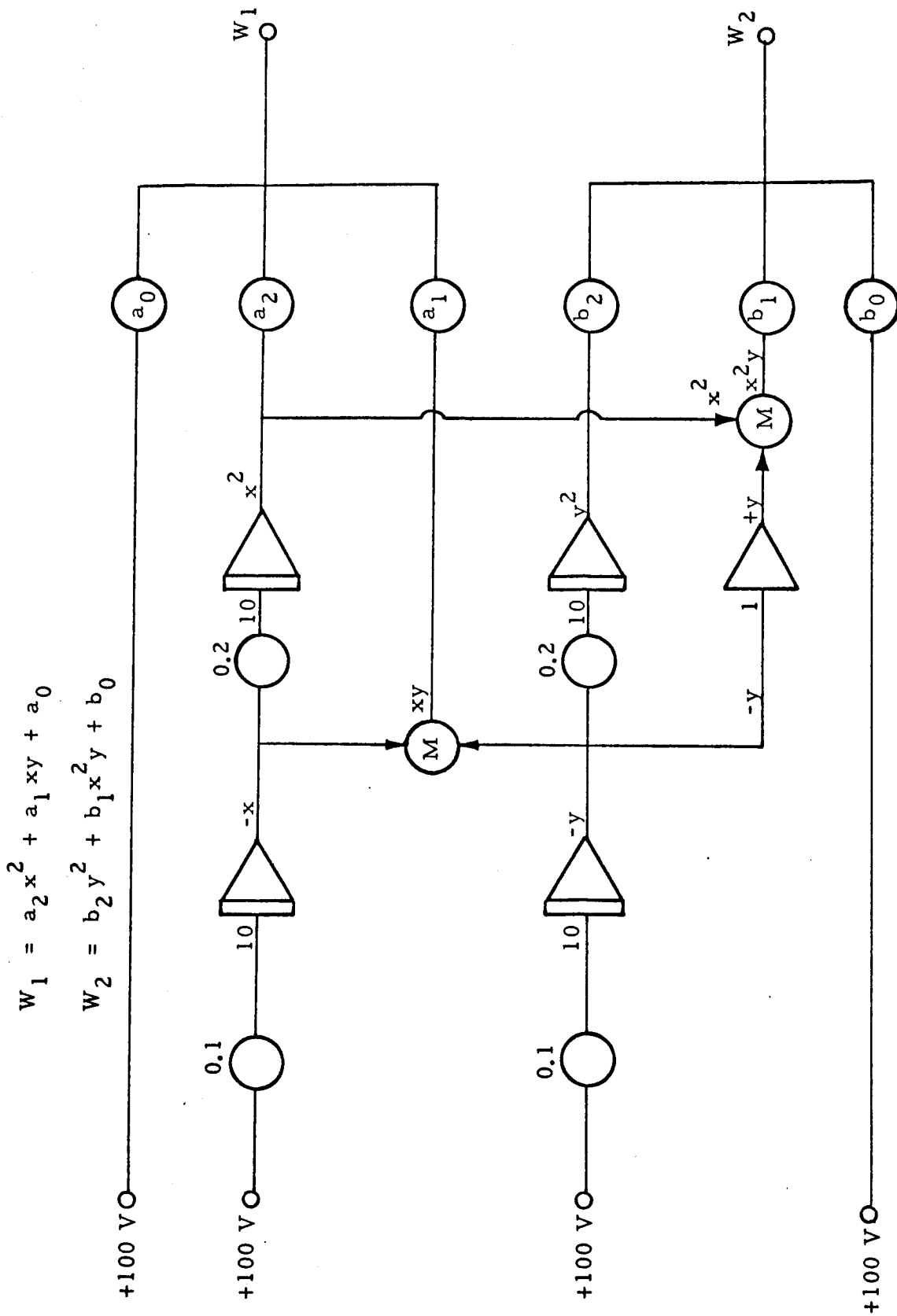


Figure 5 - Analog Solution to Two-Equation Systems

2.3.2 Hybrid Technique

Fortunately, analog and digital computers are essentially complementary in that the favorable features of one computer correspond to the undesirable features of the other. The analog computer may be said to have inherent speed due to parallel operation. Single operations of addition, for example, are much faster on the digital than the analog. However, considering a series of many operations, the analog quite often results in a lower cost in terms of computing time, since the discrete actions of the digital require separate execution of many different operations to perform the same job as the continuous acting analog.

The accuracy and resolution of the digital is far superior to the analog. The decision making capabilities, large available data storage and pre-tested subroutines and software are among the outstanding features of the digital computer. The floating-point arithmetic eliminates the scaling required by the analog.

Among the outstanding features of the analog, we should include the ability to perform simple, true integration. Unlike the discrete steps taken by the digital, the analog integrates continuously with time as the independent variable.

As has been noted in previous sections of this report, when in search of roots for nonlinear simultaneous equations, our digital routines are plagued with an absence of information concerning approximate locations of the roots. As they stand alone, all digital techniques need initial estimates. Even though they need no initial estimates for roots, our analog routines are insufficient because of their inaccuracy. A combination of the two is inevitable - hence, we select a feasible hybrid computer scheme in which the analog provides the initial estimates to the roots and the digital refines these estimates through iterations to some desired accuracy. Truitt (Reference 15) and King (Reference 16) point out many interesting arguments for hybrid

computers. But, necessity has brought the hybrid for solutions to systems of algebraic equations with the digital as a supervisor and refiner of the analog. The hybrid computer to be considered is discussed by Truitt in Reference 17. Figure 6 illustrates the logic flow of the proposed hybrid computer scheme.

Figures 7, 8, and 9 compare the digital, analog and hybrid routines with respect to percentage of solution uncertainty and required computer execution time. In all cases, we see that Fletcher-Powell seems to be superior to Newton-Raphson, and as the complexity of the system increases, this superiority becomes even greater. The hybrid routine consists of the analog integration technique and the digital modular program.

Figure 7 shows that for a four equation system, Regula-Falsi has nearly located the root before the necessary partials required by the other two digital routines have been computed. The analog solution has a rough estimate to the roots in a short time but is unable to refine the values. The hybrid, however, selects the estimates as soon as they are available from the analog and passes them on to the Regula-Falsi technique for a very fast accurate result.

In Figure 8 we see that an increase in the size of the system of equations makes the computation of partials the advantageous mode of solution. The analog accuracy is even worse now, since an increase in computing elements has brought balancing problems as well as an introduction of noise.

Figure 9 illustrates still less accuracy in the analog and increased computation time for computing the partials. The hybrid technique, once again, is preferred.

Evaluation of a hybrid technique consisting of a modular digital program and an integrator analog program reveals that favorable features of one program correspond to the undesirable features of the other program.

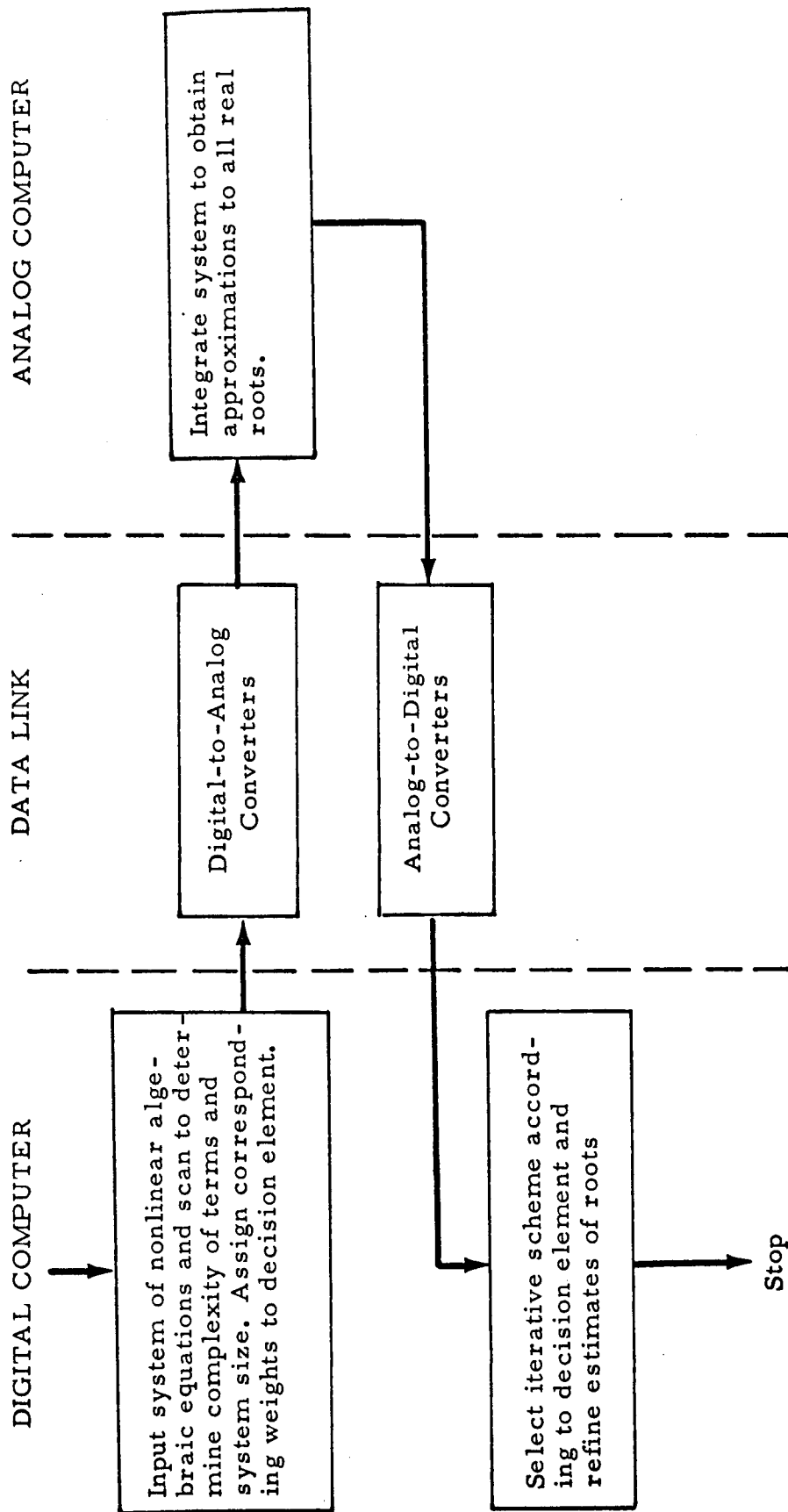


Figure 6 - Hybrid Scheme

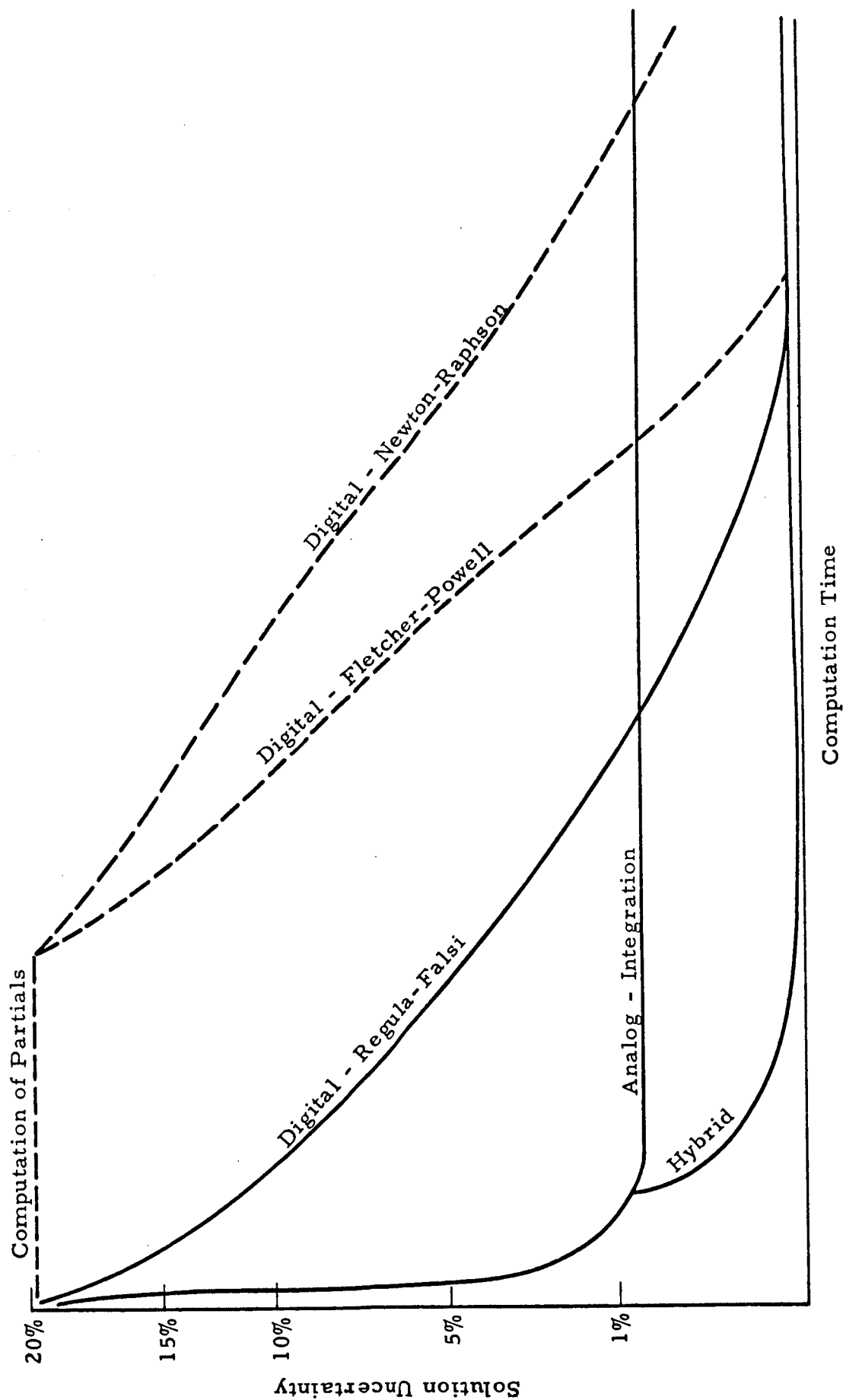


Figure 7 - Four Equation System (Algebraic Terms)

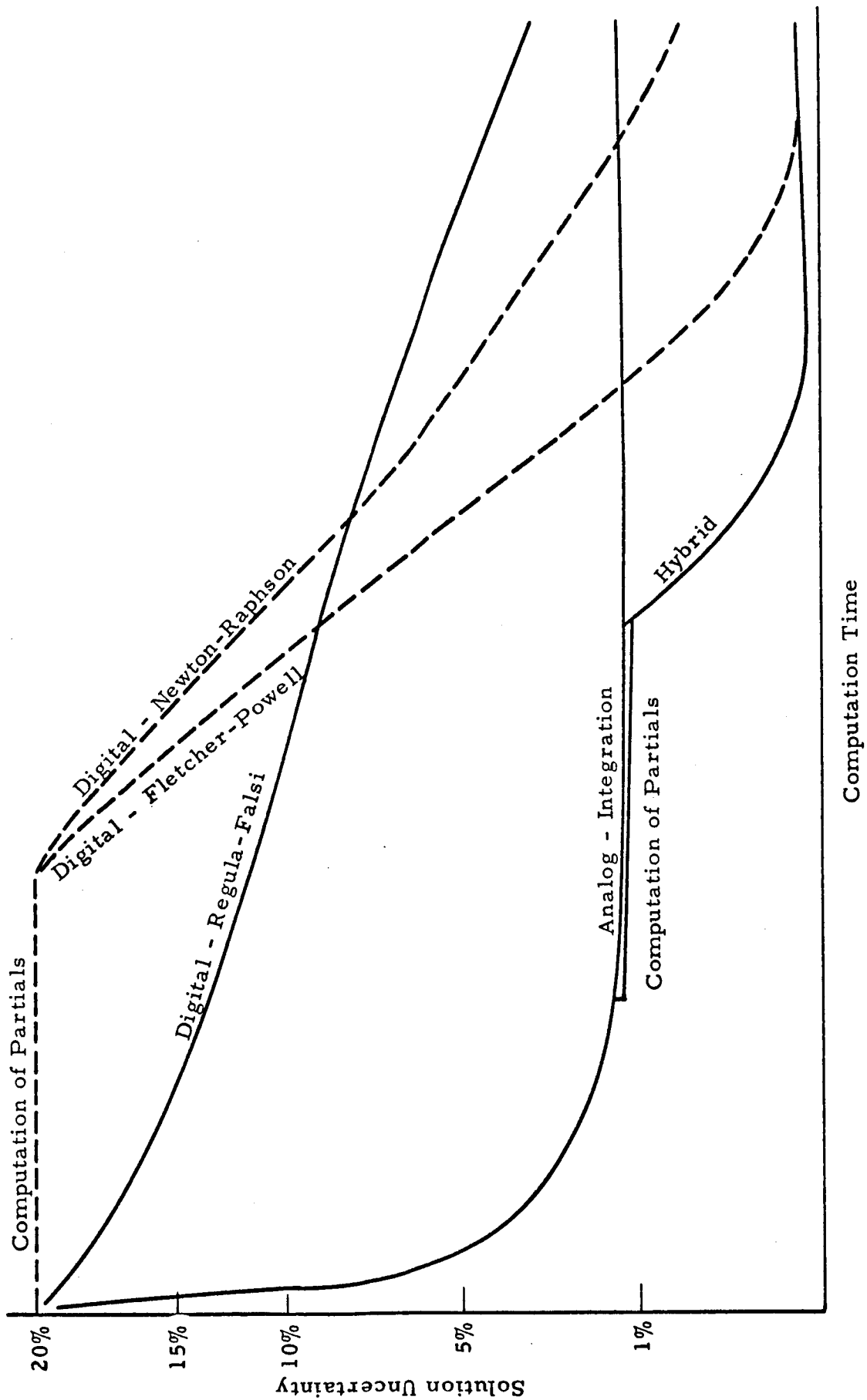


Figure 8 - Ten Equation System (Algebraic Terms)

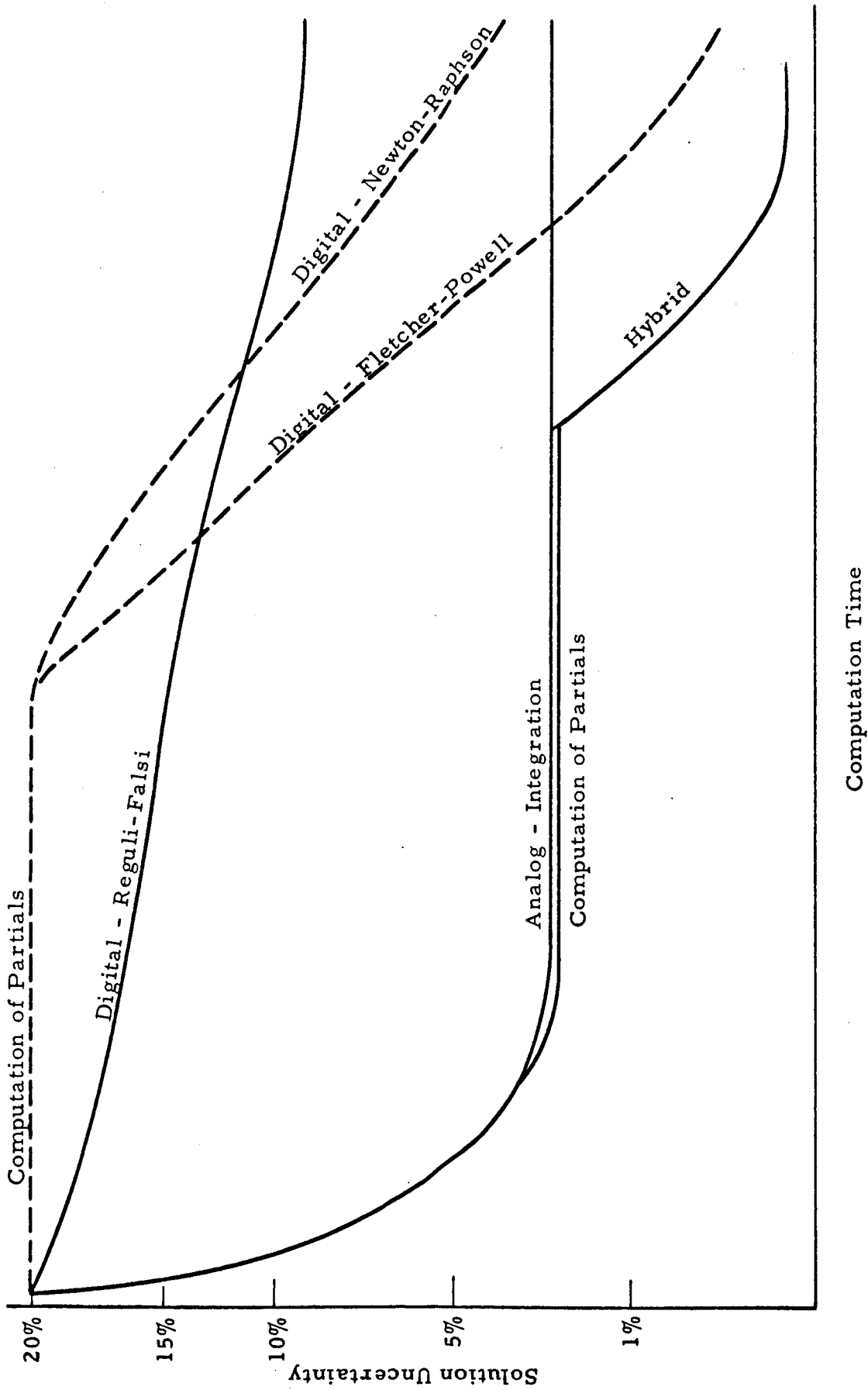


Figure 9 - Fifteen Equation System (Transcendental and Algebraic Terms)

Specifically, the need for a search routine to locate estimates of the roots for a digital program, such as those of Kiefer (Reference 5) and Newman (Reference 6), are unnecessary on the hybrid computer. Similarly, the inaccuracy of the analog is compensated for by the iterations of the digital. Our conclusions are that accuracy of solution, reliability of method for general purpose use, cost (direct result of computation time required for acceptable accuracy), and the compatibility of solutions with the physical problem giving rise to the nonlinear algebraic system would be realized in their maximum state when an analog-digital computer system is used.

2.4 TEST RESULTS AND ENGINEERING APPLICATIONS

The following discussion is presented to demonstrate the computer program, compare the different methods on the basis of some test results, and exemplify the usefulness of such a program. The first subsection considers several systems of simultaneous nonlinear equations, including one system with closely spaced roots, while the second subsection is devoted to an engineering problem of particular interest.

2.4.1 Test Results

Several systems of nonlinear algebraic equations were solved in an effort to determine the reliability and accuracy of the solution techniques under study. Following is a list of systems and results of solutions where the prescribed accuracy is $\epsilon = .00001$.

$$1. \quad \begin{cases} f_1 = x_1^2 + x_2 - 11 \\ f_2 = x_1 + x_2^2 - 7 \end{cases} \quad \text{Known solution:} \quad \begin{cases} x_1 = 3.5844283 \\ x_2 = -1.8481262 \end{cases} .$$

The Fletcher-Powell technique required four iterations in one second for

$$\begin{cases} x_1 = 3.5844283 \\ x_2 = 1.8481262 \end{cases} .$$

The Newton-Raphson technique required three iterations in one second for

$$\begin{cases} x_1 = 3.5844284 \\ x_2 = -1.8481265 \end{cases} .$$

The search routine required 39 iterations in three seconds for

$$\begin{cases} x_1 = 3.5844284 \\ x_2 = -1.8481265 \end{cases} .$$

$$2. \quad \begin{cases} f_1 = x_1^2 + 12x_2 - 1 \\ f_2 = 49x_1 + 49x_2^2 + 84x_1 + 2324x_2 - 681 \end{cases} \quad \text{Known solution:} \quad \begin{cases} x_1 = .14285 \\ x_2 = .28571 \end{cases} .$$

The Fletcher-Powell technique failed. After six iterations the routine became trapped in the minimization process. After six iterations in one second

$$\begin{cases} x_1 = .1370932 \\ x_2 = .2857944 \end{cases} .$$

The Newton-Raphson technique required 37 iterations in two seconds for

$$\begin{cases} x_1 = .14291311 \\ x_2 = .28571195 \end{cases} .$$

$$3. \quad \begin{cases} f_1 = x_1 + x_2 + x_3 - 1 \\ f_2 = 3x_1 + x_2 - 3x_3 - 5 \\ f_3 = x_1 - 2x_2 - 5x_3 - 10 \end{cases} \quad \text{Known solution:} \quad \begin{cases} x_1 = 6 \\ x_2 = -7 \\ x_3 = 2 \end{cases} .$$

The Fletcher-Powell technique required five iterations in two seconds for

$$\begin{cases} x_1 = 6 \\ x_2 = -7 \\ x_3 = 2 \end{cases} .$$

The Newton-Raphson technique required three iterations in one second for

$$\begin{cases} x_1 = 6 \\ x_2 = -7 \\ x_3 = 2 \end{cases} .$$

$$4. \quad \begin{cases} f_1 = 121x_1^2 - 32x_2^2 - 121 \\ f_2 = 7x_1^2 + 7x_1x_2 + 7x_2^2 + 70x_1 - 63x_2 - 34 \end{cases} \quad \text{Known solution:} \quad \begin{cases} x_1 = 1.2857 \\ x_2 = 1.5710 \end{cases} .$$

The Fletcher-Powell technique required fourteen iterations in one second for

$$\begin{cases} x_1 = 1.2857981 \\ x_2 = 1.5685939 \end{cases} .$$

The Newton-Raphson technique required sixteen iterations in two seconds for

$$\begin{cases} x_1 = 1.2857746 \\ x_2 = 1.5816151 \end{cases} .$$

$$5. \quad \begin{cases} f_1 = x_1^3 - 3x_1^2x_2 + x_2^2 - 7 \\ f_2 = x_1^2 - 4x_1 + x_2^2 - 4x_2 + 4 \end{cases} \quad \text{Known solution:} \quad \begin{cases} x_1 = 1.91475 \\ x_2 = .001817 \end{cases} .$$

The Fletcher-Powell technique required six iterations in two seconds for

$$\begin{cases} x_1 = 1.9147505 \\ x_2 = .0018179036 \end{cases}$$

The Newton-Raphson technique required five iterations in two seconds for

$$\begin{cases} x_1 = 1.9147503 \\ x_2 = .0018176978 \end{cases}$$

$$6. \quad \begin{cases} f_1 = x_1^2 + x_2^2 + x_3^2 - 1 \\ f_2 = 2x_1^2 + x_2^2 - 4x_3 \\ f_3 = 3x_1^2 - 4x_2 + x_3^2 \end{cases} \quad \text{Known solution: } \begin{cases} x_1 = .785202 \\ x_2 = .496611 \\ x_3 = .369922 \end{cases}$$

The Fletcher-Powell technique required seven iterations in three seconds for

$$\begin{cases} x_1 = .78519690 \\ x_2 = .49661120 \\ x_3 = .36992267 \end{cases}$$

The Newton-Raphson technique required five iterations in two seconds for

$$\begin{cases} x_1 = .78519695 \\ x_2 = .49661139 \\ x_3 = .36992283 \end{cases}$$

$$7. \quad \begin{cases} f_1 = x_1 + 10x_2 \\ f_2 = \sqrt{5}(x_3 - x_4) \\ f_3 = (x_2 - 2x_3)^2 \\ f_4 = \sqrt{10}(x_1 - x_4)^2 \end{cases} \quad \text{Known solution: } \begin{cases} x_1 = 0 \\ x_2 = 0 \\ x_3 = 0 \\ x_4 = 0 \end{cases}$$

The Fletcher-Powell technique, after fifty iterations in five seconds, obtained

$$\begin{cases} x_1 = -.49446885 \times 10^{-2} \\ x_2 = .48518575 \times 10^{-3} \\ x_3 = -.25345917 \times 10^{-2} \\ x_4 = -.25341771 \times 10^{-2} \end{cases}$$

The Newton-Raphson technique converged in 28 iterations in two seconds for

$$\begin{cases} x_1 = .14646 \times 10^{-3} \\ x_2 = -.14646 \times 10^{-4} \\ x_3 = .38937 \times 10^{-4} \\ x_4 = .38937 \times 10^{-4} \end{cases}$$

The search routine after 141 iterations requiring nine seconds obtained

$$\begin{cases} x_1 = 0 \\ x_2 = .1415963 \times 10^{-5} \\ x_3 = .8961344 \times 10^{-8} \\ x_4 = .1096351 \times 10^{-14} \end{cases}$$

$$\begin{aligned} 8. \quad & \begin{cases} f_1 = 2x_1 - 3.18309886 \log[(1-x_3)/(1+x_3)] - 1 \\ f_2 = 1.57079632x_1(1-x_2^2x_3^2)\{(x_3^2-x_2^2)/[x_3x_2^2(1+x_3^2)]\} - 5 \\ f_3 = x_3 - .14271816 \end{cases} \end{aligned}$$

$$\text{Known solution: } \begin{cases} x_1 = 0.042591338 \\ x_2 = 0.041400152 \\ x_3 = 0.14271816 \end{cases}$$

The Fletcher-Powell technique required 18 iterations in seven seconds for

$$\begin{cases} x_1 = 0.042591337 \\ x_2 = 0.041400143 \\ x_3 = 0.14271816 \end{cases}$$

The Newton-Raphson technique required 41 iterations in twenty-two seconds for

$$\begin{cases} x_1 = 0.042513292 \\ x_2 = 0.041403138 \\ x_3 = 0.14271816 \end{cases}$$

The search routine of Nelder and Mead is not used in most of the above systems because of the difficulty in deriving simplex points. Since there seemed to be difficulty in solving System 7, described above, the search routine was employed to help locate an initial estimate for one of the other iterative techniques. More time and many more iterations were required for the search routine than for any of the other techniques; however, a close-to-zero solution was obtained, which is the true solution. Therefore, because of the difficulty in obtaining simplex points and because of the length of computations, the search routine seems the best technique to be used when questions arise concerning the validity of the solution obtained by other techniques.

System 2 failed under the Fletcher-Powell technique but was successful when the Newton-Raphson technique was used. Failure of the Fletcher-Powell technique can be attributed to problems in determining the interval length along the line leading to the minimum. Davidon, in Reference 18, discusses this problem and his ideas helped to eliminate the failures of the Fletcher-Powell technique in all but one of the test cases.

Newton-Raphson has been extremely reliable in all but one (System 7) test case. The accuracy of the Newton-Raphson technique, however, has not been as good as that of Fletcher-Powell. Study of larger and more complicated systems (as shown by System 8) indicate the Fletcher-Powell technique to be more reliable than the Newton-Raphson when the system increases.

Study of System with Closely Spaced Roots

The following system of two equations in two unknowns was adopted for the purpose of analyzing a system with closely spaced roots:

$$\begin{aligned} f_1 &\equiv x^2 + y^2 - 1 = 0 \\ f_2 &\equiv x^2 - y - 1 - \epsilon = 0 \end{aligned}$$

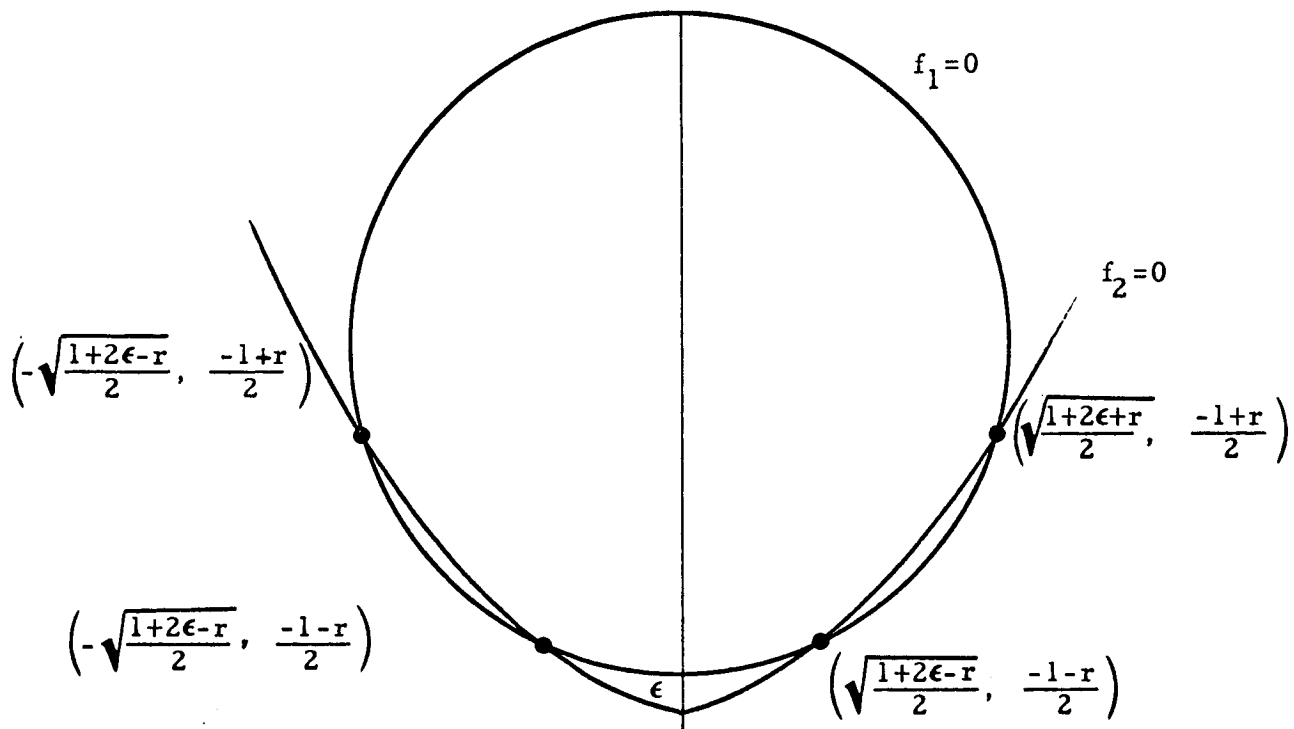


Figure 10 - Graphs of Two Equation Systems with Closely Spaced Roots

There are four distinct real roots to this system (i.e., four intersections of the circle with the parabola as shown in Figure 10) provided $0 < \epsilon < 1/4$. The spacing between the two roots at the bottom of the figure is $\sqrt{2} \sqrt{1 + 2\epsilon - r}$ where $r = \sqrt{1 - 4\epsilon}$. This spacing is approximately $2\sqrt{2\epsilon}$ for small ϵ and vanishes for $\epsilon = 0$.

The Fletcher-Powell method for solving this system determines the minimum points of the function $f_1^2 + f_2^2$, these minima being zero at the roots of the system. A contour map of $f_1^2 + f_2^2$ is shown in Figure 11 for the case when $\epsilon = 0$ (i.e., when the two bottom roots in Figure 10 coincide). Only the contours for positive x are shown, the figure being symmetrical about the y axis. Such a figure is useful for exhibiting the approximate location of the roots and the important topographic features of the function. For example, a saddle point is seen to exist at the point $(.8, -.5)$ (and likewise at $(-.8, -.5)$ by symmetry). At this point the function decreases if one proceeds in either direction along the dashed line shown, with the equation $x^2 = -1/2(y+2)(y-2)$, whereas the function increases if one proceeds from this point in a direction perpendicular to the dashed line. In any descent method one starts close to the desired root such that saddle points, etc., are avoided in the descent. A contour map of $f_1^2 + f_2^2$ for the case of a small finite value of ϵ is not available yet. Nevertheless, the Fletcher-Powell method was employed, the starting point being

$$x_0 = 0.5$$

$$y_0 = 2.0$$

After eleven iterations the value of the function was reduced to approximately 2×10^{-14} , corresponding to

$$x = .10012652$$

$$y = -.99497470$$

The method evidently sought out the lower right hand root of Figure 10, the

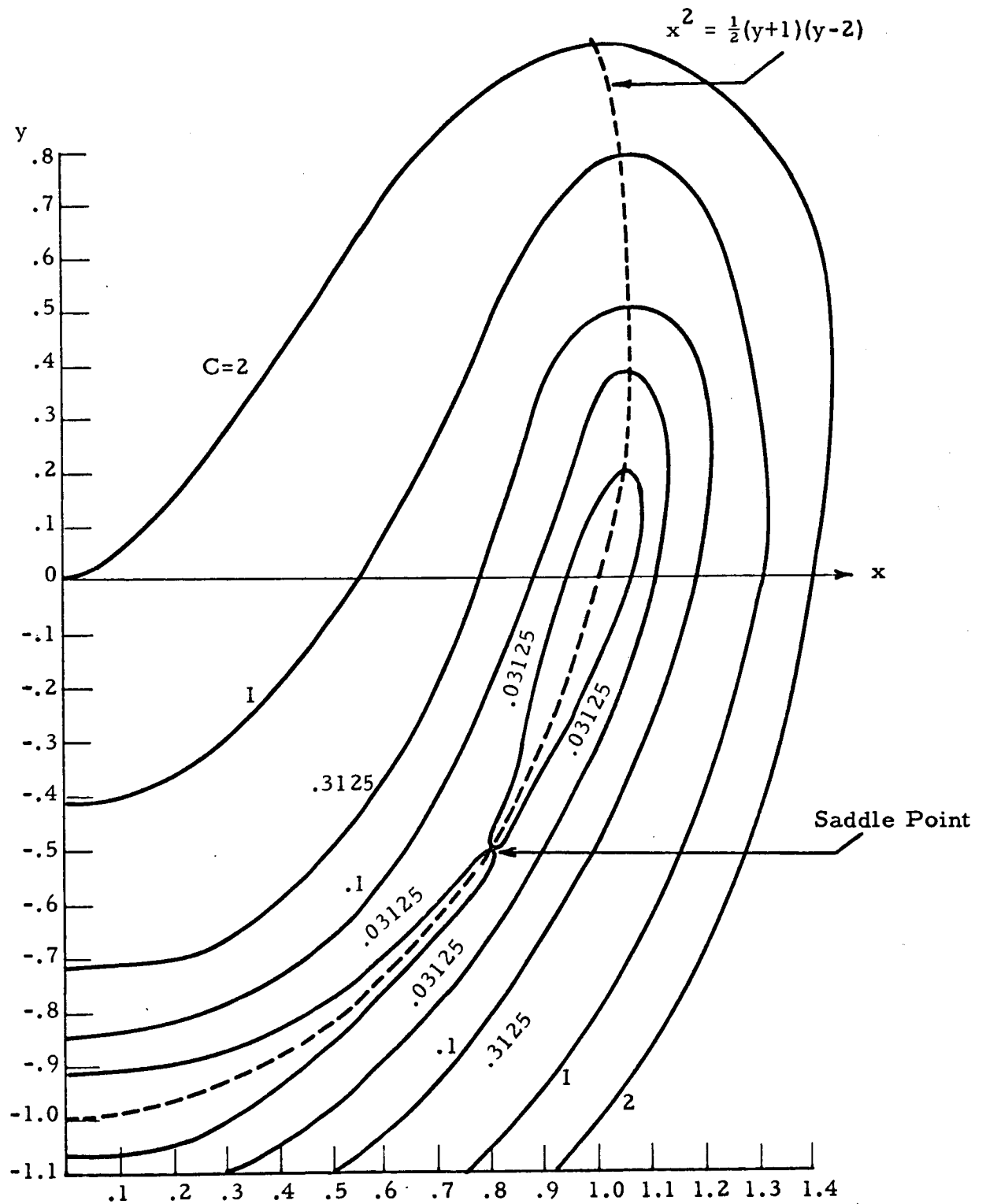


Figure 11 - Contour Plots for the Two-Equation System
with $\epsilon=0$ (Two Coincident Roots)

correct value of this root (for $\epsilon = .005$) being

$$x = +.10012619$$

$$y = -.99497475$$

The Newton-Raphson, applied directly to this system and using the same starting point gave, in eight iterations:

$$x = .10012613$$

$$y = -.99497475$$

The various iterative techniques will generally fail to distinguish between closely spaced roots unless the starting point is suitably chosen. To aid in the intelligent use of any scheme, a good knowledge of the topography of the function to be minimized is invaluable. To this end, the contour plotting routine, based upon the method of McCue, is quite useful.

2.4.2 Engineering Applications

The engineering problems in which systems of nonlinear algebraic equations arise are numerous and hence it is impossible to consider all of them. Of particular interest to the NASA/MSFC technical director of this study contract are (1) computer programs for synthesizing networks with resistive loads (Reference 19) and (2) solutions of systems of nonlinear algebraic equations that arise when some technique, such as the Ritz Averaging Method, is applied to nonlinear differential equations (Reference 1). The latter problem was considered as a basis for defining the major requirements for a computer technique.

R. S. Ryan, in an Aero-Astrodynamic Technical Note (Reference 2), describes the Ritz Method for a highly nonlinear air spring problem for both the single-degree-of-freedom and the two-degree-of-freedom situation. The former leads to a pair of simultaneous nonlinear algebraic equations in

two unknowns for the required response; the latter leads to a system of six equations in six unknowns. A detailed review of this problem follows:

Single-Degree-of-Freedom System

The application of the Ritz Averaging Method to solve nonlinear vibration problems is treated in References 1 and 2. The general second order differential equation for a single-degree-of-freedom system is

$$a \ddot{q} + b g(\dot{q}) + cf(q) = P \cos \Omega t$$

where q is the dependent variable (e.g., a displacement) $g(\dot{q})$, in the damping term, has the dimension of \dot{q} , and $f(q)$, in the restoring force term, has the dimension of q . P is the amplitude of the applied force. Klotter, in Reference 1, writes this in the form

$$E = \ddot{q} + 2DKg(\dot{q}) + K^2 f(q) - p \cos \tau = 0$$

where

$$2DK = \frac{b}{a}; K^2 = \frac{c}{a}; p = \frac{P}{a}; \tau = \Omega t$$

and assumes a periodic solution of the form

$$\bar{q} = Q \cos(\tau - \epsilon)$$

where Q and ϵ are constants to be determined. It is shown that the Ritz Averaging Method furnishes the two conditions

$$\int_0^{2\pi} E(\bar{q}) \begin{cases} \cos \tau \\ \sin \tau \end{cases} d\tau = 0$$

These conditions, when the integration is performed, lead to two nonlinear equations for the unknown quantities Q and ϵ . He then treats the undamped case ($g = 0$) for the following three restoring forces

$$(a) \quad f(q) = q$$

leading to the known solution $Q = \pm \frac{p}{K^2 - \Omega^2}$

$\epsilon = 0, \pi$ respectively.

(b) $f(q) = q + \mu^2 q^3$, $\mu^2 > 0$, the Duffing equation for a "hardening" spring.

In this case the Ritz conditions lead to

$$\frac{\Omega^2}{K^2} = 1 + \frac{3}{4} \mu^2 Q^2 + \frac{p}{K^2 Q}$$

$\epsilon = 0, \pi$ respectively

(c) $f(q) = q - \mu^2 q^3$, Duffing's equation for a "softening" spring. Here the result is

$$\frac{\Omega^2}{K^2} = 1 - \frac{3}{4} \mu^2 Q^2 + \frac{p}{K^2 Q}$$

Curves for these results are given in Figures 1 and 2 of Reference 1. The Duffing equations are also treated for linear damping, $g(q) = \dot{q}$.

In Reference 2, R. Ryan treats the single-degree-of-freedom differential equation without damping. In the notation of this reference

$$D(\eta) = \ddot{\eta} + \omega_0^2 [(1-\eta)^{-\gamma} - 1] + \Omega^2 \bar{\zeta}_0 \sin \Omega t = 0$$

where

η = normalized displacement

$\gamma = \frac{c_p}{c_v}$ ratio of specific heat at

constant pressure to that at constant volume, a constant greater than unity.

Ω , $\bar{\zeta}_0$ = frequency, normalized amplitude of applied sinusoidal force.

ω_0 = undamped natural frequency of linearized system.

An assumed solution of the form $\eta = M + Q \sin \tau$ is taken, (the additive constant appears since the restoring force is no longer symmetrical), so that the Ritz method furnishes

$$\int_0^{2\pi} D(\eta) d\tau = 0$$

$$\int_0^{2\pi} D(\eta) \sin \tau d\tau = 0$$

giving the following two equations for M and Q:

$$\int_0^{2\pi} \frac{d\tau}{(1 - M - Q \sin \tau)^\gamma} = 2\pi \quad (12)$$

$$\int_0^{2\pi} \frac{\sin \tau d\tau}{(1 - M - Q \sin \tau)^\gamma} = \pi r^2 \gamma (Q - \xi_0)$$

where $r^2 = \frac{\Omega^2}{\omega^2}$, $\omega^2 = \omega_0^2 \gamma$. Solutions (M, Q) to these equations for

$\gamma = 1.0, 1.1, 1.5, 2.0$ as obtained by analytical means and various computer techniques are derived and the results are plotted as a function of r, in Figures 12 and 13. Only the M curve is given, the Q curve being very similar.

These solutions for (12) have been obtained as follows:

a. $\gamma = 1.0$

Consider Equations (12) for $\gamma = 1.0$. The two integrals appearing

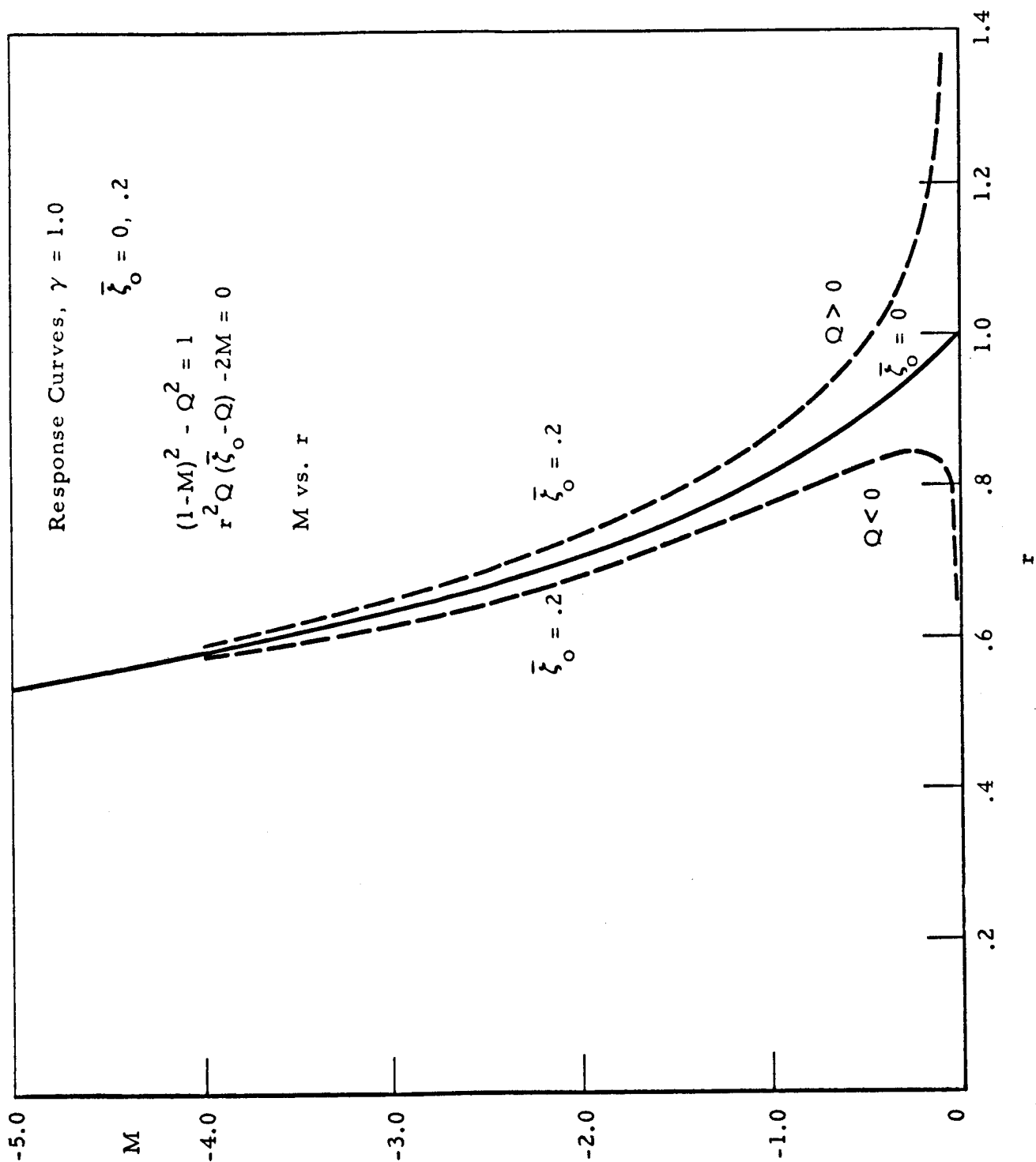


Figure 12- Response Curves for $\gamma = 1.0$

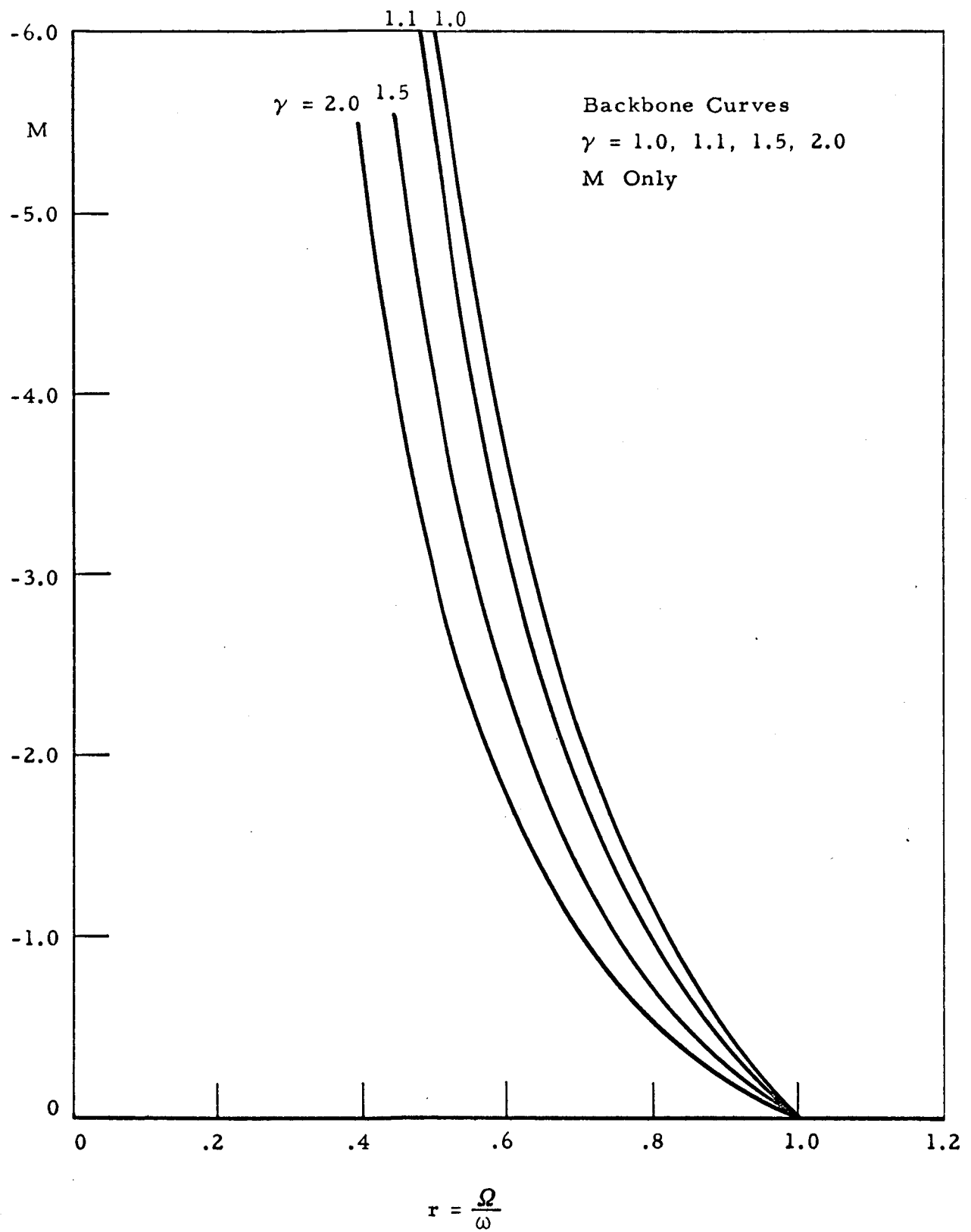


Figure 13- Backbone Curves

therein are readily evaluated.

$$\int_0^{2\pi} \frac{d\tau}{1-M-Q \sin\tau} = \frac{2\pi}{[(1-M)^2 - Q^2]^{1/2}}$$

$$\begin{aligned} \int_0^{2\pi} \frac{\sin\tau d\tau}{1-M-Q \sin\tau} &= -\frac{1}{Q} \int_0^{2\pi} \frac{1-M-Q \sin\tau - (1-M)}{1-M-Q \sin\tau} d\tau \\ &= -\frac{2\pi}{Q} \left[1 - \frac{1-M}{[(1-M)^2 - Q^2]^{1/2}} \right] \end{aligned}$$

and Equations (12) reduce to the two algebraic equations

$$(1-M)^2 - Q^2 = 1$$

$$r^2 Q(\bar{\xi}_0 - Q) - 2M = 0$$

(13)

The solution to Equation (13) for $\bar{\xi}_0 = 0$ is readily obtained:

$$M = 2 \left(1 - \frac{1}{r^2} \right), \quad Q = \pm \frac{2}{r} \sqrt{1 - r^2}$$

These results give the backbone curve. The plot M vs r is shown in Figures 12 and 13. For non-vanishing values of $\bar{\xi}_0$, one may eliminate Q , for example, from Equation (13) and obtain the response in the form r vs M as

$$r = \left[\frac{2M}{\pm \sqrt{M^2 - 2M} \bar{\xi}_0 - (M^2 - 2M)} \right]^{1/2} \quad (14)$$

Note that

$$Q = \pm \sqrt{M^2 - 2M}$$

Equation (14) is plotted for $\bar{\zeta}_0 = .2$ in Figure 12. The curve to the left of the backbone curve corresponds to $Q < 0$. The curve to the right of the backbone curve corresponds to $Q > 0$.

The Newton-Raphson program was applied to Equation (13) with $\bar{\zeta}_0 = 0, .04, .1, .2$ to determine M and Q for different values of r . Very precise results were obtained, and the roots for $\bar{\zeta}_0 = .2$ were found to agree with response curves of Figure 12.

b. $\gamma = 1.1$

Letting $c = \frac{Q}{1-M}$, the Equations (12) may be written in the form

$$1-M = \left[\frac{1}{2\pi} \int_0^{2\pi} \frac{d\tau}{(1-c \sin \tau)^\gamma} \right]^{\frac{1}{\gamma}} \quad (15)$$

$$1-M = \frac{2}{r^{2\gamma} c} \frac{\int_0^{2\pi} \frac{\sin \tau d\tau}{(1-c \sin \tau)^\gamma}}{\int_0^{2\pi} \frac{d\tau}{(1-c \sin \tau)^\gamma}} + \frac{\bar{\zeta}_0}{c}$$

These two expressions for $1-M$ were calculated for $\gamma = 1.1$ using a Runge-Kutta-Gill routine, with c as the independent variable and r as a parameter, and $\bar{\zeta}_0 = 0$. The two functions were plotted, and the intersections were accurately determined, leading to precise numbers for M and Q vs r . The backbone curve for M is plotted in Figure 13.

A simple argument for small values of M , Q in Equation (15) shows that the backbone curve touches the r axis at the point $r = 1$, for all values of γ .

c) $\gamma = 1.5$

In Equation (12) let $\gamma = \frac{3}{2}$, and put $k^2 = \frac{-2Q}{1-M-Q}$, $Q < 0$. Then the two equations become

$$\frac{4k^3 E(k)}{(-2Q)^{3/2}(1-k^2)} = 2\pi \quad (16)$$

$$\frac{8k K(k)}{(-2Q)^{3/2}} + 2\pi \left(1 - \frac{2}{k^2}\right) + \pi r^2 \gamma (\bar{\xi}_0 - Q) = 0$$

where E and K are the complete elliptic integrals

$$E = \int_0^{\frac{\pi}{2}} \sqrt{1 - k^2 \sin^2 x} dx$$

$$K = \int_0^{\frac{\pi}{2}} \frac{dx}{\sqrt{1 - k^2 \sin^2 x}}$$

From Equation (16) and the definition of k^2 , and using $\gamma = \frac{3}{2}$

$$Q = -\frac{1}{2} \left(\frac{2}{\pi}\right)^{2/3} \frac{k^2 E(k)^{2/3}}{(1-k^2)^{2/3}} \quad (17)$$

$$M = 1 + Q \left(\frac{2}{k^2} - 1\right)$$

$$r^2 = -\frac{2}{3} \frac{1}{(\bar{\xi}_0 - Q)} \frac{k^2 E(k) \left(1 - \frac{2}{k^2}\right) + 2K(k) (1-k^2)}{k^2 E(k)}$$

For various values of k , a computer program determined Q , M , r^2 from Equation (17) and plotted M and Q versus r . The M curve is shown in Figure 13. Again $\bar{\xi}_0 = 0$.

d) $\gamma = 2$

When $\gamma = 2$, the two integrals appearing in Equations (12) are evaluated as

$$\frac{2\pi (1-M)}{[(1-M)^2 - Q^2]^{3/2}}, \quad \frac{2\pi Q}{[(1-M)^2 - Q^2]^{3/2}}$$

respectively, so that Equations (12) lead to the pair of algebraic equations

$$[(1-M)^2 - Q^2]^{3/2} = 1-M \quad (18)$$

$$r^2 (\bar{\xi}_0 - Q) + \frac{Q}{1-M} = 0$$

These equations are readily solved when $\bar{\xi}_0 = 0$, yielding

$$M = 1 - \frac{1}{r^2} \quad (19)$$

$$Q = \pm \frac{1}{r^2} [1 - r^{4/3}]^{1/2}$$

The M response curve is plotted in Figure 13. Observe from Equation (19) that $M = 0$ when $r = 1$, confirming what was stated earlier about the intercept at $r = 1$. Also, values of $r > 1$ are not allowed, for otherwise Q is imaginary.

Inasmuch as the integrals in (12) cannot, in general, be expressed in closed form, the approach adopted in Reference 2 is to employ a polynomial approximation to the restoring force. Specifically a fifth degree polynomial

$$P_5(\eta) = \sum_{i=1}^5 a_i \eta^i, \text{ with } a_1 = 1, \text{ approximating } (1-\eta)^{-\gamma} - 1 \text{ in a least square}$$

sense in the interval $-2 \leq \eta \leq .7$ is employed. Plots of the restoring force and the corresponding polynomial fit, as developed at HREC are presented in Figures 14, 15, 16, and 17 for $\gamma = 1.1, 1.2, 1.3, 1.4$. The numerical values of the coefficients a_i are also shown. The polynomial fit permits the $(1-M-Q \sin \tau)^{-\gamma}$ factor in (12) to be written as a polynomial in M and Q . The integrals in (12) may then be evaluated as polynomials in M and Q , and the Equations (12) reduce to the following pair of algebraic equations:

$$\begin{aligned} 8 P_5(M) + 4 P_3(M) Q^2 + 3 P_1(M) Q^4 &= 0 \\ r^2 \gamma (\bar{\xi}_0 - Q) + P_4(M) Q + \frac{3}{4} P_2(M) Q^2 + \frac{5}{8} a_5 Q^5 &= 0 \end{aligned} \quad (20)$$

where

$$P_5(M) = \sum_{i=1}^5 a_i M^i$$

$$P_4(M) = a_1 + 2a_2 M + 3a_3 M^2 + 4a_4 M^3 + 5a_5 M^4 = P_5'(M)$$

$$P_3(M) = a_2 + 3a_3 M + 6a_4 M^2 + 10a_5 M^3 = \frac{1}{2} P_5''(M)$$

$$P_2(M) = a_3 + 4a_4 M + 10a_5 M^2 = \frac{1}{6} P_5'''(M)$$

$$P_1(M) = a_4 + 5a_5 M = \frac{1}{24} P_5^{IV}(M)$$

Equations (20) are written out in full in Reference 2. To make this report as complete as possible, they are repeated here as follows:

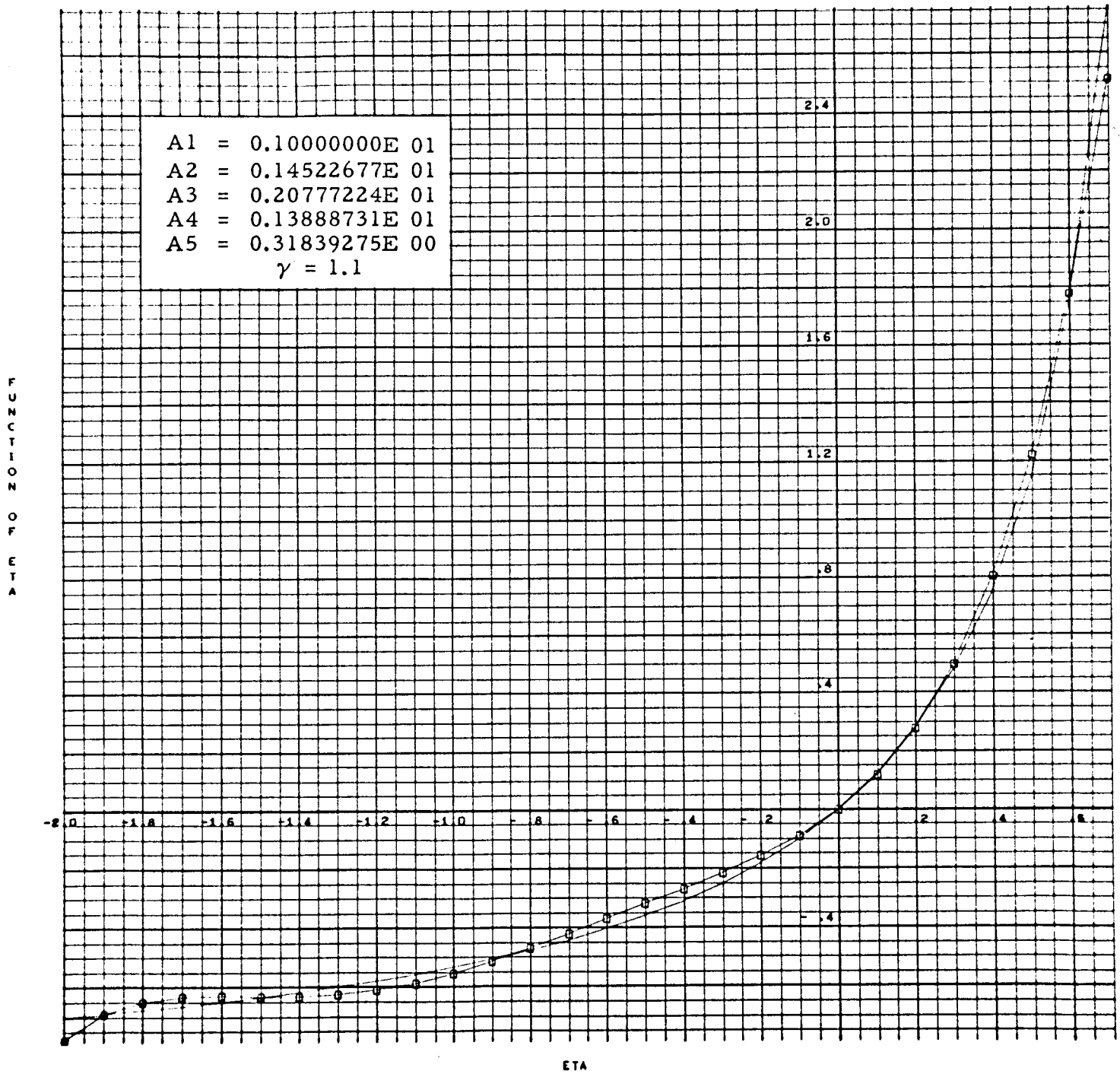


Figure 14 - Restoring Force $(1-\eta)^{-\gamma} - 1$ and Polynomial Fit, for $\gamma = 1.1$

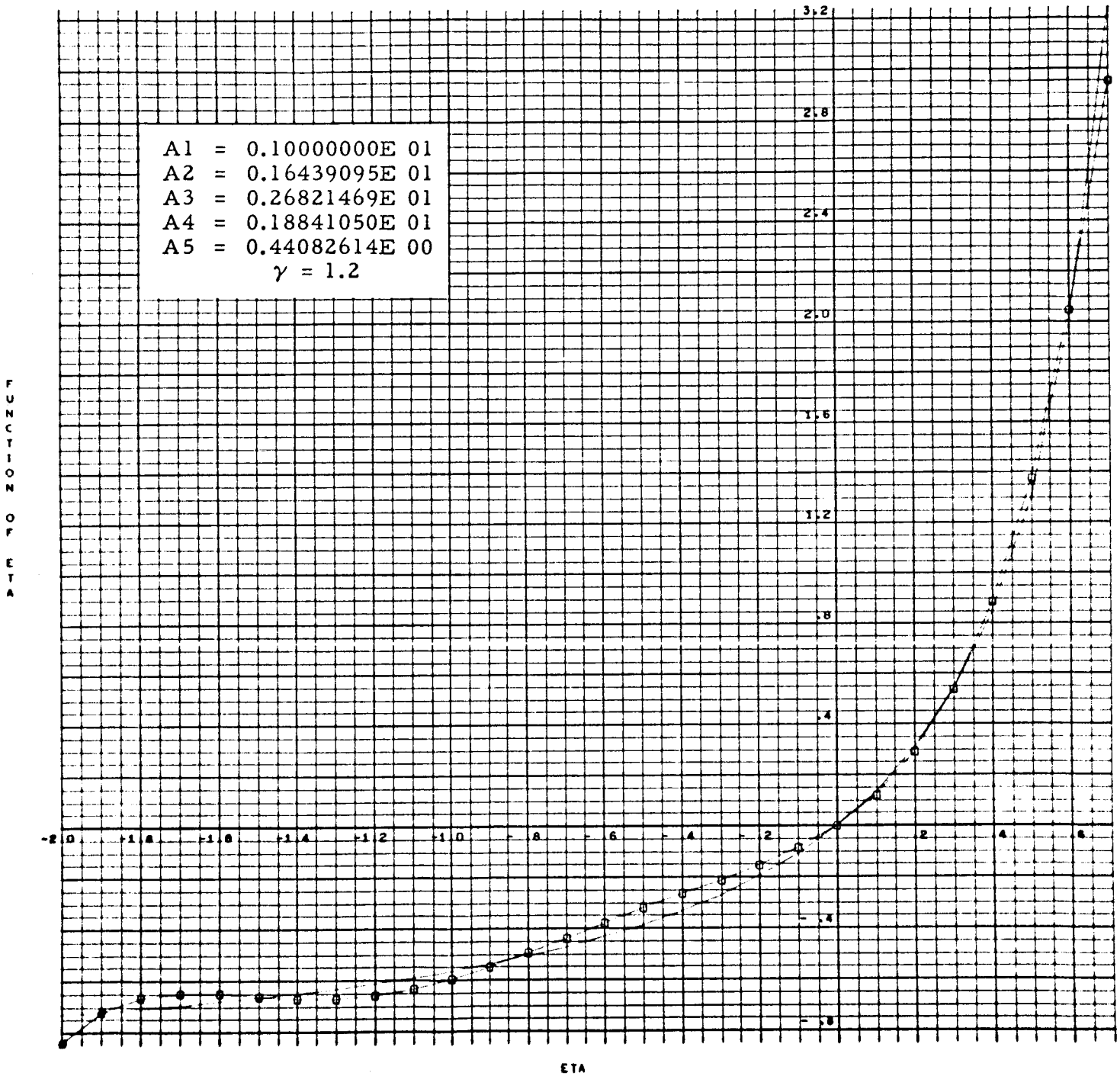


Figure 15 - Restoring Force $(1-\eta)^{-\gamma} - 1$ and Polynomial Fit, for $\gamma = 1.2$

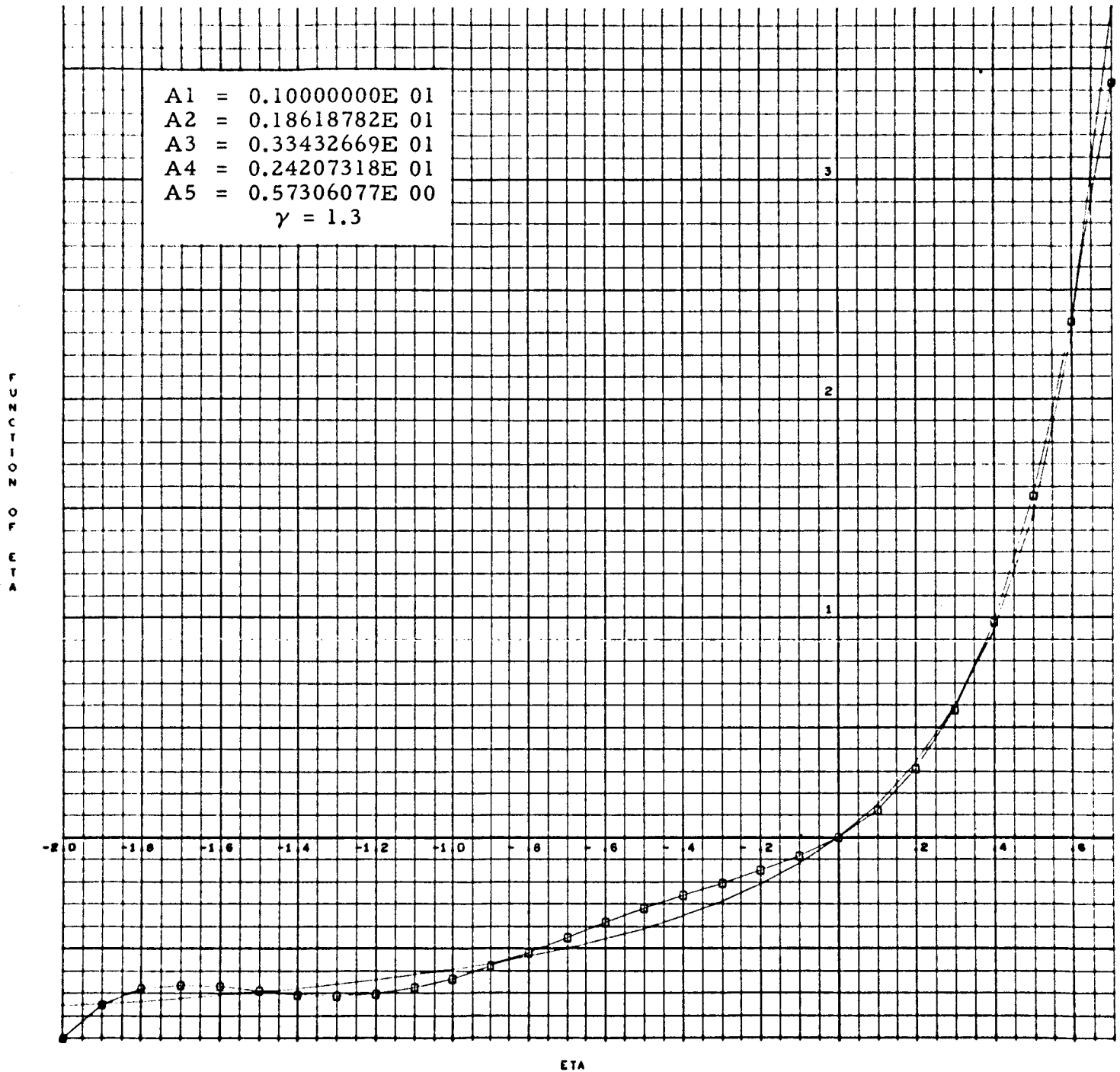


Figure 16 - Restoring Force $(1-\eta)^{-\gamma} - 1$ and Polynomial Fit, for $\gamma = 1.3$

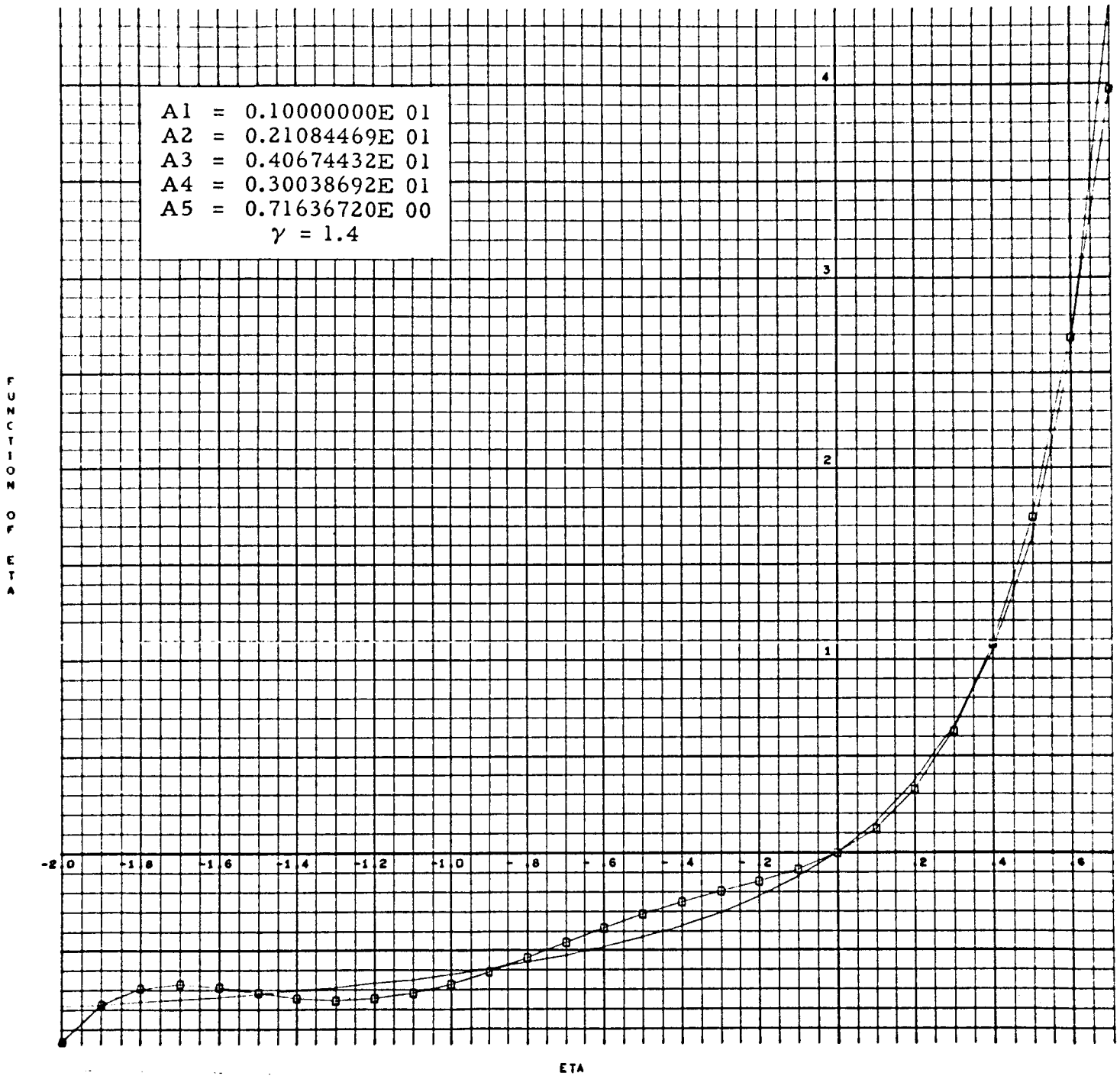


Figure 17 - Restoring Force $(1-\eta)^{-\gamma} - 1$ and Polynomial Fit, for $\gamma = 1.4$

System of two equations in two unknowns:

$$\begin{aligned}
& 8(a_1 M + a_2 M^2 + a_3 M^3 + a_4 M^4 + a_5 M^5) + 4(a_2 + 3a_3 M + 6a_4 M^2 + 10a_5 M^3)Q^2 \\
& + 3(a_4 + 5a_5 M)Q^4 = 0 \\
& -r^2(-\bar{\xi}_0 + Q)\gamma + (a_1 + 2a_2 M + 3a_3 M^2 + 4a_4 M^3 + 5a_5 M^4)Q + \frac{3}{4}(a_3 + 4a_4 M \\
& + 10a_5 M^2)Q^3 + \frac{5}{8}a_5 Q^5 = 0
\end{aligned}$$

These algebraic equations were solved by the method of steepest descent by R. Ryan, and the response curves, showing M and Q versus r, are to be found in Reference 2. His result for M versus r for $\gamma = 1.1$ is given in Figure 18. Within the aforementioned limitations on the polynomial fit, these results compare favorably with those obtained analytically from Equation (12) as explained in the previous paragraph.

Two-Degree-of-Freedom System

A two-degree-of-freedom system, involving the same nonlinear restoring force, is discussed in Reference 2. The following pair of differential equations, in the dependent variables Z_1, Z_2 is obtained.

$$\begin{aligned}
D_i = A_{i1}\ddot{Z}_1 + A_{i2}\ddot{Z}_2 + \frac{\bar{K}_{i1}}{\gamma} \left[(1-Z_1)^{-\gamma} - 1 \right] + \frac{\bar{K}_{i2}}{\gamma} \left[(1-Z_2)^{-\gamma} - 1 \right] \\
+ \Omega^2 \bar{\xi}_0 \left[B_i \sin \tau + E_i \cos \tau \right] = 0, \quad i = 1, 2
\end{aligned} \tag{21}$$

where the coefficients are defined in Reference 2. Assuming a solution of the form

$$Z_i = M_i + Q_i \cos \tau + R_i \sin \tau, \quad i = 1, 2 \tag{22}$$

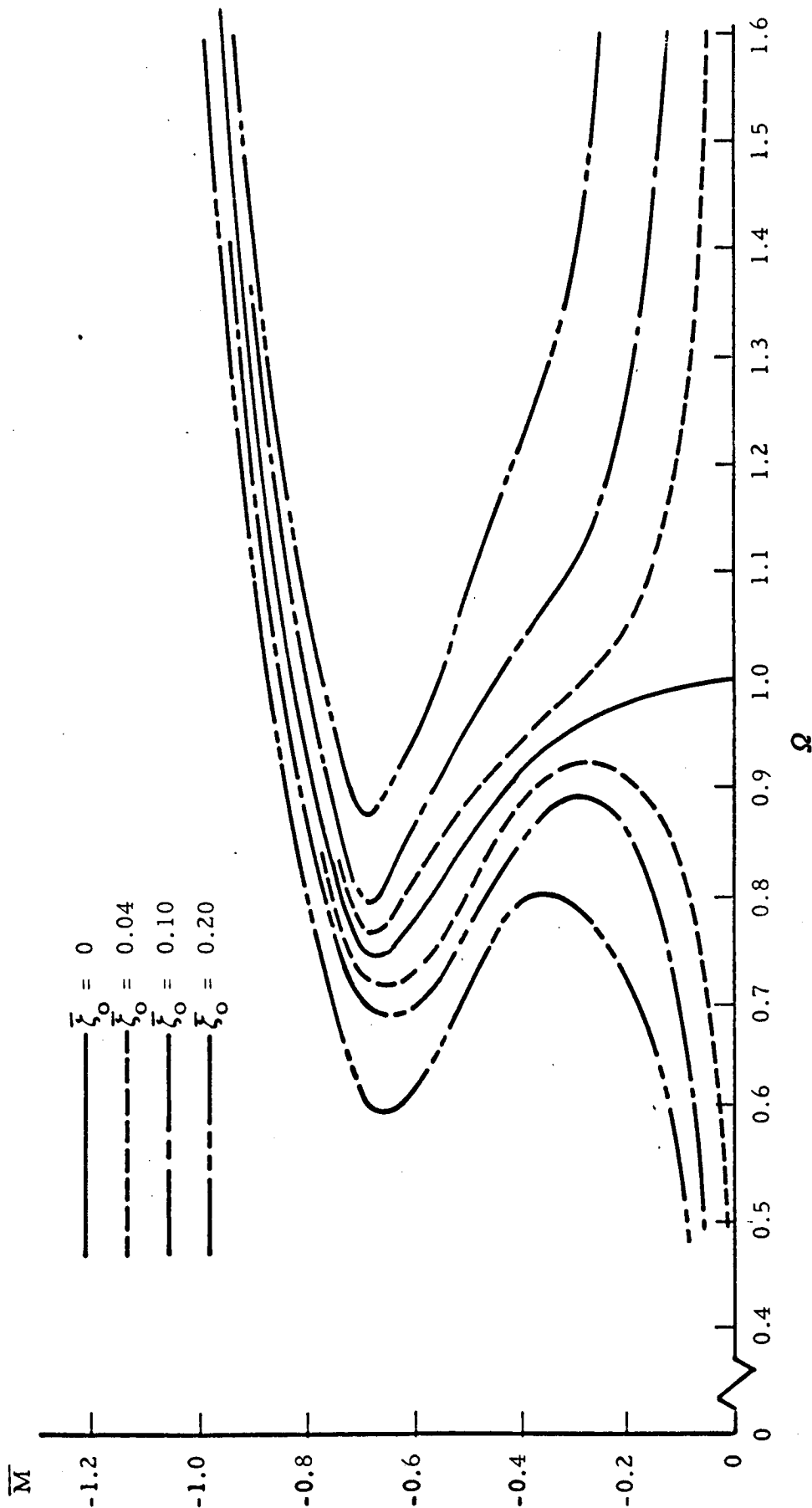


Figure 18 - Single Degree of Freedom Response $\gamma = 1.1$, From Reference 2 Obtained By Curve Fit To Restoring Force

where M_i , Q_i , R_i are to be determined, the Ritz method requires that the six relationships

$$\int_0^{2\pi} D_i(Z_i) \begin{cases} 1 \\ \cos \tau \\ \sin \tau \end{cases} d\tau = 0, \quad i = 1, 2 \quad (23)$$

must hold.

Substituting Equation (22) into (21) and carrying out the integration in Equation (23), the result is:

$$\frac{\bar{K}_{11}}{\gamma} \int_0^{2\pi} f_1 d\tau + \frac{\bar{K}_{12}}{\gamma} \int_0^{2\pi} f_2 d\tau = \frac{2\pi}{\gamma} (\bar{K}_{11} + \bar{K}_{12})$$

$$\frac{\bar{K}_{21}}{\gamma} \int_0^{2\pi} f_1 d\tau + \frac{\bar{K}_{22}}{\gamma} \int_0^{2\pi} f_2 d\tau = \frac{2\pi}{\gamma} (\bar{K}_{21} + \bar{K}_{22})$$

$$-A_{11}\Omega^2 Q_1\pi - A_{12}\Omega^2 Q_2\pi + \frac{\bar{K}_{11}}{\gamma} \int_0^{2\pi} f_1 \cos \tau d\tau + \frac{\bar{K}_{12}}{\gamma} \int_0^{2\pi} f_2 \cos \tau d\tau = 0$$

$$-A_{21}\Omega^2 Q_1\pi - A_{22}\Omega^2 Q_2\pi + \frac{\bar{K}_{21}}{\gamma} \int_0^{2\pi} f_1 \cos \tau d\tau + \frac{\bar{K}_{22}}{\gamma} \int_0^{2\pi} f_2 \cos \tau d\tau = 0$$

$$-A_{11}\Omega^2 R_1\pi - A_{12}\Omega^2 R_2\pi + \frac{\bar{K}_{11}}{\gamma} \int_0^{2\pi} f_1 \sin \tau d\tau + \frac{\bar{K}_{12}}{\gamma} \int_0^{2\pi} f_2 \sin \tau d\tau = 0$$

$$-A_{21}\Omega^2 R_1\pi - A_{22}\Omega^2 R_2\pi + \frac{\bar{K}_{21}}{\gamma} \int_0^{2\pi} f_1 \sin\tau d\tau + \frac{\bar{K}_{22}}{\gamma} \int_0^{2\pi} f_2 \sin\tau d\tau = 0$$

Using the abbreviations

$$J_i = \int_0^{2\pi} f_i d\tau$$

$$C_i = \int_0^{2\pi} f_i \cos\tau d\tau \quad (24)$$

$$S_i = \int_0^{2\pi} f_i \sin\tau d\tau$$

where

$$f_i = (1 - M_i - Q_i \cos\tau - R_i \sin\tau)^{-\gamma}$$

the six-equation system may be written

$$\bar{K}_{11}J_1 + \bar{K}_{12}J_2 = 2\pi (\bar{K}_{11} + \bar{K}_{12}) \quad (25a)$$

$$\bar{K}_{21}J_1 + \bar{K}_{22}J_2 = 2\pi (\bar{K}_{21} + \bar{K}_{22}) \quad (25b)$$

$$-\Omega^2 \pi (A_{11}Q_1 + A_{12}Q_2) + \frac{\bar{K}_{11}}{\gamma} C_1 + \frac{\bar{K}_{12}}{\gamma} C_2 = -\Omega^2 \bar{\xi}_0 E_1 \pi \quad (25c)$$

$$-\Omega^2 \pi (A_{21}Q_1 + A_{22}Q_2) + \frac{\bar{K}_{21}}{\gamma} C_1 + \frac{\bar{K}_{22}}{\gamma} C_2 = -\Omega^2 \bar{\xi}_0 E_2 \pi \quad (25d)$$

$$-\Omega^2 \pi (A_{11}R_1 + A_{12}Q_2) + \frac{\bar{K}_{11}}{\gamma} S_1 + \frac{\bar{K}_{12}}{\gamma} S_2 = -\Omega^2 \bar{\xi}_0 B_1 \pi \quad (25e)$$

$$-\Omega^2 \pi (A_{21}R_1 + A_{22}R_2) + \frac{\bar{K}_{21}}{\gamma} S_1 + \frac{\bar{K}_{22}}{\gamma} S_2 = -\Omega^2 \bar{\xi}_0 B_2 \pi \quad (25f)$$

The first two of these equations imply immediately

$$J_1 = J_2 = 2\pi \quad (26)$$

When $\gamma = 1.0$, the integrals (24) may be evaluated as

$$\begin{aligned} J_i &= 2\pi \left[(1-M_i)^2 - Q_i^2 - R_i^2 \right]^{-1/2} \\ C_i &= \frac{2\pi Q_i}{R_i^2 + Q_i^2} \left[-1 + (1-M_i) \left\{ (1-M_i)^2 - Q_i^2 - R_i^2 \right\}^{-1/2} \right] \quad (27) \\ S_i &= \frac{R_i}{Q_i} C_i \end{aligned}$$

(see, for example, Reference 20, Table 69, No. 17).

Hence, for $\gamma = 1.0$, Equation (20) implies

$$(1-M_i)^2 - Q_i^2 - R_i^2 = 1, \quad i = 1, 2 \quad (28)$$

Consider the case $\bar{\xi}_0 = 0$ (This gives the "backbone" curve), with $\gamma = 1.0$. In this case it is possible to reduce the system (25) to a system of three equations in three unknowns. Eliminating C_2 from the third and fourth equations,

$$\begin{aligned} \Omega^2 \pi \left[(\bar{K}_{12} A_{21} - \bar{K}_{22} A_{11}) Q_1 + (\bar{K}_{12} A_{22} - \bar{K}_{22} A_{12}) Q_2 \right] \\ + \frac{1}{\gamma} (\bar{K}_{11} \bar{K}_{22} - \bar{K}_{21} \bar{K}_{12}) C_1 = 0 \quad (29) \end{aligned}$$

and S_2 from the fifth and sixth,

$$\begin{aligned} \Omega^2 \pi \left[(\bar{K}_{12} A_{21} - \bar{K}_{22} A_{11}) R_1 + (\bar{K}_{12} A_{22} - \bar{K}_{22} A_{12}) R_2 \right] \\ + \frac{1}{\gamma} (\bar{K}_{11} \bar{K}_{22} - \bar{K}_{21} \bar{K}_{12}) S_1 = 0 \end{aligned} \quad (30)$$

and then multiplying (29) by $\frac{S_1}{C_1} = \frac{R_1}{Q_1}$ and subtracting (30), it is observed that

$$Q_1 R_2 - R_1 Q_2 = 0 \quad (31)$$

In view of (27), (31), it follows that

$$C_1 = \frac{2\pi Q_1}{R_1^2 + Q_1^2} \left[-1 + (1 + R_1^2 + Q_1^2)^{1/2} \right], \quad S_1 = \frac{R_1}{Q_1} C_1 \quad (32)$$

$$C_2 = \frac{2\pi Q_1 R_1}{(R_1^2 + Q_1^2) R_2} \left[-1 + \left\{ 1 + \frac{R_2^2}{R_1^2} (R_1^2 + Q_1^2) \right\}^{1/2} \right], \quad S_2 = \frac{R_1}{Q_1} C_2$$

Rewrite (30), (25e) and (25f),

$$\Omega^2 \pi \left[(\bar{K}_{12} A_{21} - \bar{K}_{22} A_{11}) R_1 + (\bar{K}_{12} A_{22} - \bar{K}_{22} A_{12}) R_2 \right] + \frac{1}{\gamma} (\bar{K}_{11} \bar{K}_{22} - \bar{K}_{22} \bar{K}_{12}) = 0$$

$$\Omega^2 \pi (A_{11} R_1 + A_{12} R_2) - \frac{\bar{K}_{11}}{\gamma} S_1 - \frac{\bar{K}_{12}}{\gamma} S_2 = 0 \quad (33)$$

$$\Omega^2 \pi (A_{21} R_1 + A_{22} R_2) - \frac{\bar{K}_{21}}{\gamma} S_1 - \frac{\bar{K}_{22}}{\gamma} S_2 = 0$$

With the above equations for S_i , the three equations (33) are a system of three equations for the unknowns R_1 , R_2 , Q_1 . After solving these, Q_2 is found from (31) and M_1 , M_2 from (28). Thus a precise solution to the problem for the case $\gamma = 1.0$ is available.

For general values of γ , the integrals S_i , C_i appearing in the system (25) cannot be written in closed form. If the polynomial fit to the restoring force $P_5(\eta) \approx (1 - \eta)^{-\gamma} - 1$ is invoked, these integrals are readily evaluated, and the system (25) reduces to six algebraic equations for M_i , Q_i , R_i . These equations are written out in full in Reference 2 as Equations (84-89), however, to make this report as complete as possible, they are repeated here as follows:

System of six equations in six unknowns:

$$\sum_{i=1}^2 \bar{K}_{1i} \left[2M_i + 2M_i^2 a_2 + 2M_i^3 a_3 + 2M_i^4 a_4 + 2M_i^5 a_5 + a_2 Q_i^2 + a_2 R_i^2 + 3a_3 M_i Q_i^2 + 3a_3 M_i R_i^2 + \frac{3}{4} a_4 Q_i^4 + \frac{3}{4} a_4 R_i^4 + 6a_4 M_i^2 Q_i^2 + 6a_4 M_i^2 R_i^2 + \frac{6}{4} a_4 Q_i^2 R_i^2 + \frac{15}{4} a_5 M_i Q_i^4 + \frac{15}{4} a_5 M_i R_i^4 + 10a_5 M_i^3 Q_i^2 + 10a_5 M_i^3 R_i^2 + \frac{30}{4} a_5 M_i Q_i^2 R_i^2 \right] = 0$$

$$\sum_{i=1}^2 K_{2i} \left[2M_i + 2a_2 M_i^2 + 2a_3 M_i^3 + 2a_4 M_i^4 + 2a_5 M_i^5 + a_2 Q_i^2 + a_2 R_i^2 + 3a_3 M_i Q_i^2 + 3a_3 M_i R_i^2 + \frac{3}{4} a_4 Q_i^4 + \frac{3}{4} a_4 R_i^4 + 6a_4 M_i^2 Q_i^2 + 6a_4 M_i^2 R_i^2 + \frac{6}{4} a_4 Q_i^2 R_i^2 + \frac{15}{4} a_5 M_i Q_i^4 + \frac{15}{4} a_5 M_i R_i^4 + 10a_5 M_i^3 Q_i^2 + 10a_5 M_i^3 R_i^2 + \frac{30}{4} a_5 M_i Q_i^2 R_i^2 \right] = 0$$

$$\Omega^2 \left[- \sum_{i=1}^2 A_{1i} Q_i + \zeta_o (\bar{b} + \bar{a} \cos \alpha) + \sum_{i=1}^2 \bar{K}_{1i} \left[Q_i + 2a_2 M_i Q_i + \frac{3}{4} a_3 Q_i^3 \right. \right. \\ \left. \left. + 3a_3 M_i^2 Q_i + \frac{3}{4} a_3 Q_i R_i^2 + 3a_4 M_i Q_i^3 + 4a_4 M_i^3 Q_i + 3a_4 M_i Q_i R_i^2 \right. \right. \\ \left. \left. + \frac{15}{24} a_5 Q_i^5 + \frac{30}{4} a_5 M_i^2 Q_i^3 + 5a_5 M_i^4 Q_i + \frac{30}{4} a_5 M_i^2 Q_i R_i^2 + \frac{5}{16} a_5 Q_i R_i^4 \right. \right. \\ \left. \left. + \frac{10}{16} a_5 Q_i^3 R_i^2 \right] = 0.$$

$$\Omega^2 \left[- \sum_{i=1}^2 A_{2i} Q_i + \bar{\zeta}_o (1 - \cos \alpha) + \sum_{i=1}^2 \bar{K}_{2i} \left[Q_i + 2a_2 M_i Q_i + \frac{3}{4} a_3 Q_i^3 \right. \right. \\ \left. \left. + 3a_3 M_i^2 Q_i + \frac{3}{4} a_3 Q_i R_i^2 + 3a_4 M_i Q_i^3 + 4a_4 M_i^3 Q_i + 3a_4 M_i Q_i R_i^2 \right. \right. \\ \left. \left. + \frac{15}{24} a_5 Q_i^5 + \frac{30}{4} a_5 M_i^2 Q_i^2 + 5a_5 M_i^4 Q_i + \frac{30}{4} a_5 M_i^2 Q_i R_i^2 + \frac{5}{16} a_5 Q_i R_i^4 \right. \right. \\ \left. \left. + \frac{10}{16} a_5 Q_i^3 R_i^2 \right] = 0.$$

$$\Omega^2 \left[- \sum_{i=1}^2 A_{1i} R_i + \bar{a} \bar{\zeta}_o \sin \alpha \right] + \sum \bar{K}_{1i} \left[R_i + 2a_2 M_i R_i + \frac{3}{4} a_3 R_i^3 \right. \\ \left. + 3a_3 M_i^2 R_i + \frac{3}{4} a_3 Q_i^2 R_i + 3a_4 M_i R_i^3 + 4a_4 M_i^3 R_i + 3a_4 M_i Q_i^2 R_i \right. \\ \left. + \frac{15}{24} a_5 R_i^5 + \frac{30}{4} a_5 M_i^2 R_i^3 + 5a_5 M_i^4 R_i + \frac{30}{4} a_5 M_i^2 Q_i^2 R_i + \frac{10}{16} a_5 Q_i^2 R_i^3 \right. \\ \left. + \frac{5}{16} a_5 Q_i^4 R_i \right] = 0.$$

$$\begin{aligned}
Q^2 \left[- \sum_{i=1}^2 A_{2i} R_i - \bar{\zeta}_0 \sin \sigma \right] + \sum \bar{K}_{2i} \left[R_i + 2a_2 M_i R_i + \frac{3}{4} a_3 R_i^3 \right. \\
+ 3a_3 M_i^2 R_i + \frac{3}{4} a_3 Q_i^2 R_i + 3a_4 M_i R_i^3 + 4a_4 M_i^3 R_i + 3a_4 M_i Q_i^2 R_i \\
+ \frac{15}{24} a_5 R_i^5 + \frac{30}{4} a_5 M_i^2 R_i^3 + 5a_5 M_i^4 R_i + \frac{30}{4} a_5 M_i^2 Q_i^2 R_i + \frac{10}{16} a_5 Q_i^2 R_i^3 \\
\left. + \frac{5}{16} a_5 Q_i^4 R_i \right] = 0.
\end{aligned}$$

All of the known methods for solving systems of nonlinear equations require prior knowledge of the approximate location of the roots of interest. In the two equation system discussed earlier, such knowledge is available from the known analytical solution for $\gamma = 1.0$ discussed at that point and presented in Reference 21. The solutions for $\gamma = 1.0$ form a good starting point for obtaining solutions for $\gamma = 1.1$, for example.

Section 3

RECOMMENDATIONS AND CONCLUSIONS

This report is as complete and self-contained as possible. It includes all of the significant content previously documented in the bi-monthly progress reports, References 21 through 27, as well as the final result. This final result is a composite computer program for solving an arbitrary system of simultaneous algebraic and transcendental equations. The term composite is employed to account for the fact that the program is a combination of techniques. These techniques are called the Newton-Raphson, Fletcher-Powell, Simplex and Contour Mapping. Such a program provides a spectrum of different approaches for seeking a solution and hence the capability for solving widely diverse classes of equations.

In applying this program to particular problems, it may quite often be useful to use one technique as an aid to another. For example, the simplex method could be used in some cases to determine a sufficiently close estimate of the root for either the Newton-Raphson or Fletcher-Powell method. The contour mapping routine may be used to obtain useful information regarding approximate location of the roots, division between closely spaced roots, number of roots, ridges, valleys and saddle points. Such knowledge is often required in order to avoid convergence problems in applying any of the other three techniques.

In addition to solving systems of nonlinear equations, another useful application of this program is in minimization (maximization) problems. The Fletcher-Powell, Simplex Method, and Contour Mapping methods are basically minimization techniques.

The principal conclusions of this study, regarding convergence problems, separating and identifying roots are:

1. The selection of a particular method for a particular system of equations is frequently essential.
2. An initial guess for the solution is always essential, and quite often it must be sufficiently close.

It is therefore recommended that future study be directed largely toward techniques for obtaining useful information which would provide a firm basis for selecting a particular method and an initial guess. This information should include the number of roots, approximate location of roots, division between closely spaced roots, and certain features of the function's contour. These methods could be developed as companion routines to the present computer program developed under this contract.

Possible approaches to the problem include: (1) automatic contour mappings, (2) extension of Cauchy's Integral Theorem, and (3) application of the Sturmian Sequence Rule. The first approach has been included in the present program, however, only a rudimentary technique was used. The decision regarding the necessity of such complimentary information was not made until late in the study and of course this was not the principal objective of the present contract. Therefore, further development of the contour mapping approach should yield some valuable results. The second and third approaches are discussed briefly by Hochstrasser. (See Item 17 of the Bibliography, Appendix A.) In developing techniques for applying any of these approaches to higher order systems of equations, the principal considerations should be computer requirements and tractability.

Hybrid computation schemes also appear to be an area in which future study would yield some significant improvements. The attractive features offered by modern analog and hybrid facilities, such as patchable digital logic and on-line displays, are particularly amenable to the gradient and search methods. In fact, these same features would be quite useful in obtaining the preliminary knowledge about a particular system of equations that is frequently essential.

In considering the extent to which the composite computer program encompasses the complete spectrum of basically different methods, there is only one method that has been excluded. This is a random search method. Such procedures were not included because they are more amenable to high-speed, repetitive analog facilities than the digital facility for which the present program was developed. Such a method would normally be preferred only if all other methods fail. It is of particular value when the gradient cannot be determined, however, in this case, the simplex search should be a satisfactory alternative.

Section 4
REFERENCES

1. Klotter, K., "Nonlinear Vibration Problems Treated by the Averaging Method of W. Ritz," Proceedings of the First National Congress of Applied Mechanics, Published by The American Society of Mechanical Engineers, 29 W. 39th St., New York 18, N. Y.
2. Ryan, Robert S., "Nonlinear Two-Degree-of-Freedom Response with Sinusoidal Inputs," Aero-Astroynamics Internal Note 13-64, MSFC, Huntsville, Alabama.
3. Hanry, D. and F. Bernede, "Solving a Set of Implicit Simultaneous Equations," Submitted by B. Farudet, SHARE Program Catalog SDA 3195, September 1964.
4. Fletcher, R. and M. J. D. Powell, "A Rapidly Convergent Descent Method for Minimization," COMP. J., Vol. 6, 1963, pp. 163-168.
5. Kiefer, J., "Sequential Minimax Search for a Maximum," Proc. Am. Math. Soc. 4, 1953, p. 502.
6. Newman, D. J., "Location of the Maximum on Unimodal Surfaces, Journal of the Assoc. for Computing Machinery, Vol. 12, No. 3, July 1965, p. 395.
7. Hooke, Robert and T. A. Jeeves, " 'Direct Search' Solution of Numerical and Statistical Problems," Journal of the Assoc. for Computing Machinery, Vol. 8, 1961, p. 212.
8. Nelder, J. A. and R. Mead, "A Simplex Method for Function Minimization," The Computer Journal, Vol. 7, 1964, p. 308.
9. Kolmogorov and Fomin, Functional Analysis, Graylock Press, Rochester, New York, 1957, Vol. 1, p. 71.
10. McCue, G. S., "Optimization and Visualization of Functions," AIAA Journal, Vol. 2, No. 1, January 1964, p. 99.
11. Huskey, Harry D. and G. A. Korn, Computer Handbook, McGraw-Hill Book Co., New York, 1962, pp. 5-139 through 5-149.
12. Karplus, W. J. and W. W. Soroka, Analog Methods Computation and Simulation, McGraw-Hill Book Company, New York, 1959, pp. 231, 260.
13. Johnson, Clarence L., Analog Computer Techniques, McGraw-Hill Book Co., New York, 1956, pp. 166-168.

14. Gephart, L., "Algebraic Systems and the REAC, Mathematical Tables and Other Aids to Computation," The National Research Council, Washington, D.C., July 1952, p. 203.
15. Truitt, T. D., "Hybrid Computation ... What Is It? ... Who Needs It?," Proceedings - Spring Joint Computer Conference, 1964, pp. 249-269.
16. King, E. M. and R. Gelman, "Experience with Hybrid Computation," Proceedings - Fall Joint Computer Conference, 1962, pp. 36-43.
17. Truitt, T. D., "A Discussion of the EAI Approach to Hybrid Computations," Simulation, October 1965, pp. 248-257.
18. Davidon, W. C., "Variable Metric Method for Minimization," Argonne National Laboratory, Lemont, Illinois, May 1959.
19. Szuchy, N. C., J. J. Lane and J. C. Johnson, "Booster Attitude Stabilization Network Synthesis," Summary Report No. 2, RAC 373-2, Contract NAS8-5016, Republic Aviation Corporation, Farmingdale, L. I., N. Y.
20. Bierens deHaan, Nouvelles Tables d'Intégrales Définies, Hafner Publishing Co., New York, 1957.
21. Cawood, D. W., "Systems of Nonlinear Equations, July-August Progress Report," LMSC/HREC A712215, Contract NAS8-20178, August 1965.
22. Lockheed Missiles & Space Company, Huntsville Research & Engineering Center, "Solution of Nonlinear Equations," Progress Report for September-October 1967, LMSC/HREC A712454, Contract NAS8-20178, November 1965.
23. Lockheed Missiles & Space Company, Huntsville Research & Engineering Center, "Systems of Nonlinear Equations," Summary Report, LMSC/HREC A712704, Contract NAS8-20178, 31 December 1965.
24. Lockheed Missiles & Space Company, Huntsville Research & Engineering Center, "Solution of Systems of Nonlinear Equations January - February Progress Report," LMSC/HREC A782234, Contract NAS8-20178, March 1966.
25. Lockheed Missiles & Space Company, Huntsville Research & Engineering Center, "Solutions of Systems of Nonlinear Equations," March-April Progress Report, LMSC/HREC A782773, Contract NAS8-20178, May 1966.
26. Lockheed Missiles & Space Company, Huntsville Research & Engineering Center, "Solutions of Systems of Nonlinear Equations," May-June Progress Report, LMSC/HREC A782961, Contract NAS8-20178, July 1966.
27. Lockheed Missiles & Space Company, Huntsville Research & Engineering Center, "Solutions of Systems of Nonlinear Equations," July-August Progress Report, LMSC/HREC A783258, Contract NAS8-20178, September 1966.

APPENDIX A

BIBLIOGRAPHY

1. Barnes, J. G. P., "An Algorithm for Solving Non-Linear Equations Based on the Secant Method," Computer Journal, April 1965, pp. 66-72.
2. Berman, G., "Minimization by Successive Approximation," SIAM Numer. Analysis, Vol. 3, No. 1, 1966, pp. 123-133.
3. Booth, A. D., "An Application of the Method of Steepest Descents to the Solution of Systems of Non-linear Simultaneous Equations," Quart. Journ. Mech. and Applied Math., Vol. 11, Part 4, pp. 460-468.
4. Brooks, S. H., "A Discussion of Random Methods for Seeking Maxima," J. Operations Res., Vol. 6, March-April, 1958, pp. 244-251.
5. Brooks, S. H., "A Comparison of Maximum Seeking Methods," J. Operations Res. Soc., Vol. 7, 1959, pp. 430-457.
6. Broyden, C. G., "A Class of Methods for Solving Nonlinear Simultaneous Equations," Mathematics of Computation, Vol. 19, No. 12, Published by American Mathematical Society, October 1965.
7. Crockett, J. G., and H. Chernoff, "Gradient Methods of Maximization," Pacific J. Math., Vol. 5, 1955, pp. 33-50.
8. Curry, H. D., "The Method of Steepest Descent for Non-Linear Minimization Problems," Quar. App. Math., Vol. 2, 1944, p. 258.
9. Davidon, W. C., "Variable Metric Method for Minimization," A. E. C. Research and Development Report, ANL-5990 (Rev.), 1959.
10. Favreau, R. R., and R. G. E. Franks, "Statistical Optimization," Second International Conference for Analog Computation, Strasbourg, France, September 1958.
11. Favreau, R. R., and R. Franks, "Random Optimization by Analog Techniques," Proc. Second Inter. Analogue Computation Meeting, Strasbourg, Presses Academiques Europeenes Brussels, 1959, pp. 437-443.
12. Fletcher, R., "Function Minimization without Evaluating Derivatives," Computer Journal, Vol. 8, April 1965, pp. 33-41.
13. Fletcher, R., and M. J. D. Powell, "Rapidly Convergent Descent Method for Minimization," Computer Journal, Vol. 6, July 1963, pp. 163-168.
14. Fletcher, R., and C. M. Reeves, "Function Minimization by Conjugate Gradients," Computer Journal, Vol. 7, July 1964, pp. 149-154.

15. Freudenstein, F., and B. Roth, "Numerical Solution of Systems of Non-Linear Equations," Journal of the A. C. M., Vol. 10, 1963, p. 550.
16. Himsworth, F. R., W. Spendley, and G. R. Hext, "The Sequential Application of Simplex Designs in Optimization and Evolutionary Operation," Technometrics, Vol. 4, p. 441.
17. Hochstrasser, V., "Numerical Methods for Finding Solutions of Non-linear Equations," Survey of Numerical Analysis, ed., J. Todd, McGraw-Hill, New York, 1962, Chap. 7.
18. Hooke, R., and T. A. Jeeves, "Direct Search Solution of Numerical and Statistical Problems," Journal of A.C.M., Vol. 8, April 1962, pp. 212-229.
19. Householder, A. S., Principles of Numerical Analysis, McGraw-Hill, New York, 1953, Chap. 3.
20. Johnson, S., "Best Exploration for Maximum is Fibonaccian," Rpt. RM-1590, Rand Corporation, Santa Monica, California.
21. Kiefer, J., "Sequential Minimax Search for a Maximum," Proc. Amer. Math. Soc. 4, 1953, pp. 502-506.
22. Kiefer, J., "Optimum Sequential Search and Approximation Methods under Minimum Regularity Assumptions," SIAM Journal, Vol. 5, 1957, pp. 105-136.
23. Kizner, W., "A Numerical Method for Finding Solutions of Nonlinear Equations," J. Soc. Indust. Appl. Math., Vol. 12, No. 2, June 1964, pp. 424-428.
24. Levine, L., Methods for Solving Engineering Problems, McGraw-Hill Book Company, New York, 1964, Chap. 8, pp. 217-255.
25. McCue, G. A., "Visualization of Functions by Stereographic Techniques," SID 63-170, North American Aviation, Inc., 20 January 1963.
26. McCue, G. A., "Optimization by Function Contouring Techniques," SID 63-171, North American Aviation, Inc., 10 February 1963.
27. McCue, Gary A., "Optimization and Visualization of Functions," AIAA Journal, Vol. 2, No. 1, January 1964, p. 99.
28. Mitchell, B. A., "A Hybrid Analog-Digital Parameter Optimizer for ASTRAC II," Simulation, Vol. 4, June 1965, pp. 399-411.
29. Munson, J. K., and A. I. Rubin, "Optimization by Random Search on the Analog Computer," IRE Transactions on Electronic Computers, June 1959, pp. 200-203.

30. Nelder, J. A., and R. Mead, "A Simplex Method for Function Minimization," Computer Journal, Vol. 7, January 1965, pp. 308-313.
31. Newman, D. J., "Location of the Maximum on Unimodal Surfaces," Journal of the Assoc. for Comp. Mach., Vol. 12, No. 3, July 1965, pp. 395-398.
32. Ostrowski, A. M., Solution of Equations and Systems of Equations, Academic Press, New York, 1960.
33. Powell, M. J. D., "An Iterative Method for Finding Stationary Values of a Function of Several Variables," Computer Journal, Vol. 5, 1962.
34. Powell, M. J. D., "A Method for Minimizing a Sum of Squares of Non-Linear Functions without Calculating Derivatives," Computer Journal, Vol. 7, January 1965, 303-307.
35. Rosenbrock, H. H., "An Automatic Method for Finding the Greatest or Least Value of a Function," Computer Journal, Vol. 3, October 1960, pp. 174-184.
36. Saaty, T. L., and Bram, J., Nonlinear Mathematics, McGraw-Hill Book Company, New York, 1964, Chap. 2, pp. 53-92.
37. Shah, B. V., R. J. Buehler, and O. Kempthorne, "The Method of Parallel Tangents (Partan) for Finding an Optimum," Office of Naval Research Report, NR-042-207 (No. 2), 1961.
38. Spang, III, H. A., "A Review of Minimization Techniques for Non-linear Functions," SIAM Review, Vol. 4, No. 4, October 1962, pp. 343-364.
39. Swann, W. H., "Report on the Development of a New Direct Search Method of Optimization," I. C. I. Ltd, Central Instrument Laboratory Research Note 64/3, 1964.
40. Witsenhausen, H. S., "Hybrid Techniques Applied to Optimization Problems," Proc. SJCC, Simulation, Fall 1963.
41. Wolfe, P., "The Secant Method for Simultaneous Non-Linear Equations," Comm. Assoc. Comp. Mach., Vol. 2, p. 12.

APPENDIX B-1

COMPOSITE PROGRAM USER'S MANUAL

There are two types of input to the composite computer program for solving systems of nonlinear algebraic equations. The first type of input consists of the system of equations to be solved, while the second type is made up of pertinent option selectors and required initial information concerning the system to be solved by the Nonlinear Equations Program (NEP).

SYSTEM INPUT

The system of equations to be solved is input directly into the program deck in the subroutine entitled, EVAL. (See Figure 1.) The user must punch the system on 80 column punch cards in the Fortran IV compiler language using Columns 7 - 72 and place the cards in their proper location in the Subroutine EVAL. If the equation should require more columns, a number (1 - 9) is placed in Column 6 and the previous card continued. All variables such as x, y, z, . . . , must be re-named X(1) for x, X(2) for y, X(3) for z, . . . , and all functions such as f, g, h, . . . , must be re-named F(1) for f, F(2) for g, F(3) for h, As an example, consider the system

$$f(x, y, z) = x + y + z - 1$$

$$g(x, y, z) = 3x + y - 3z - 5$$

$$h(x, y, z) = x - 2y - 5z - 10$$

which would be written for inclusion in the Subroutine EVAL as

$$F(1) = X(1) + X(2) + X(3) - 1.$$

$$F(2) = 3. * X(1) - X(2) - 3. * X(3) - 5.$$

$$F(3) = X(1) - 2. * X(2) - 5. * X(3) - 10.$$

The maximum number of equations allowable is 20. Although the gradient must be defined analytically when using Feltcher-Powell or Newton-Raphson, the user need not derive the equations since the program computes numerical partials using the functions as input. (See the sample run for an example of proper system inclusion.)

OPTION AND DATA INPUT

The MAVRIK input package is utilized for data input from punch cards. All desired input parameters are placed on 80-column punch cards beginning in Column 2 and extending through Column 72 with additional cards added as needed to complete data input requirements. A slash mark (/) indicates the end of input data.

All input parameters have been given names or symbols and are assigned numerical values by equating the symbols to the values desired; e.g., METHOD = 3, POINTS = 21., 3.4, 5., creates input data for METHOD equal to 3, and the first three variables of array entitled POINTS equal to 21., 3.4, and 5. (POINTS (1,1) = 21., POINTS (2,1) = 3.4, POINTS (3,1) = 5.) All arrays are assigned values by columns, hence the innermost subscript varies first of the doubly subscripted variables. To skip to later variables in the array, one can write POINTS + 2 = 20., 13., which assigns values to POINTS (3,1) and POINTS (4,1). To skip to variables in the second column of the array, one must add the number of rows in the first column in order to get to the second column. For example, POINTS is dimensioned (21, 20) so the first element of the second column (POINTS (1,2)) is POINTS + 21 and POINTS (2,2) is POINTS + 22.

Input data is in the floating point format and input options are in the fixed or integer format. A decimal point must be included in all floating point fields. A comma must follow each numerical value regardless of format. Blanks may be placed throughout the input data for ease in reading as long as symbols and numerical fields do not contain blanks among their elements; e.g., 805 not 80 5, XL not X L. However, XL = 805., POINTS = 25., 31., 65., 21., 23., are

acceptable. Note that 28.3 , and 5 , are treated as 28.30 and 50, respectively. If an integer 5 were desired, the comma must follow the 5 with no blanks (5,).

Input Options

There are four input options the user may select. Three methods of solution and one contour plotting technique are available.

Option Selector

<u>Method</u>	<u>Technique Selector (Integer Variable)</u>
1	means use the Fletcher-Powell technique
2	means use the Newton-Raphson technique
3	means use the search routine
4	means construct contour plots

Input Data

The selection of certain methods require specific additional information. The letters following the input data symbols indicate which options require the data in question: (F) Fletcher-Powell, (N) Newton-Raphson, (S) search routine and (C) contour plots.

UNKNOWN (F, N, S, C)	=	N where N is the number of unknowns the system has (an integer variable).
ORDER (F)	=	K where K is the order of the system (an integer variable).
XINITL (F, N)	=	X_N where X_N is the array of initial estimates for all unknowns (floating point variables dimensioned (20)).
DEL (F, N)	=	ΔX where ΔX is the increment to be used in computing partial derivatives of the system with respect to each unknown (floating point variable). Set to .001 if not input.

EPS (F, N, S)	=	ϵ where ϵ is the accuracy criterion for termination of iterative techniques (floating point variable). Set to .00001 if not input.
POINTS (S)	=	P_{ij} where P_{ij} are simplex points and $i - 1 = j$ is the size of the system (floating point variables dimensioned (21, 20)).
ALPHA (S)	=	α where α is a reflection coefficient (floating point variable).
BETA (S)	=	β where β is a contraction coefficient (floating point variable).
GAMMA (S)	=	γ where γ is an expansion coefficient (floating point variable).
ITEST (F, N, S)	=	J where J is the maximum number of iterations to be allowed in solving for roots (integer variable). Set to 50 if not input.
XL (C)	=	x_l where x_l is the left most x value for grid construction (floating point).
XR (C)	=	x_r where x_r is the right most x value for grid construction (floating point).
YT (C)	=	y_t where y_t is the top most y value for grid construction (floating point).
YB (C)	=	y_b where y_b is the bottom most y value for grid construction (floating point).
N (C)	=	n where n is the number of contour plots desired (integer variable).
SF (C)	=	scale factor when unequal increments for the contour values are desired. (floating point)

SAMPLE COMPUTER RUN

$$F(X, Y, Z) = X^2 + Y^2 + Z^2 - 1.$$

$$G(X, Y, Z) = 2X^2 + Y^2 - 4Z$$

$$H(X, Y, Z) = 3X^2 - 4Y + Z^2$$

Program Listing B-1 shows the necessary conversion of the above system to Fortran IV and proper inclusion into the Subroutine EVAL. Program Listing B-2 illustrates the input data cards for solving the system.

Program Listing B-3 shows the intermediate output as a result of rising the Newton-Raphson solution technique.

Program Listing B-4 shows the solution to the system.

PROGRAM LISTING B-1

LMSCJS

C69B

- EFN SOURCE STATEMENT - IFN(S) -

```

SUBROUTINE EVAL(X,FX2,GX2,PAR)
COMMON /INIT/ IS,IN,DEL, EPS , METHOD,ITER
DIMENSION X(20),F(20),GX2(20) ,PAR(20,20),DX(20)
DIMENSION F1(20), DF1(20)
IF(METHOD.EQ. 4) GO TO 10
ICOUNT=1
IFLAG=C
GO TO 10
5 CONTINUE
X(ICOUNT)=X(ICOUNT)+DEL
10 CONTINUE
C WRITE IN SYSTEM BEGINNING IN COLUMN 7 WITH VARIABLES AS X(I) SUCH AS
C F(1)=X(1)**2+X(2)-11.
C F(2)=X(2)**2+X(1)-7.
C FOR THE SYSTEM
C      X**2+Y-11.=0.
C      Y**2+X- 7.=0.
C WHERE
C X=X(1)
C Y=X(2)
C
C PLACE SYSTEM BETWEEN HERE
C F(1)=X(1)**2+X(2)**2+X(3)**2-1.
C F(2)=2.*X(1)**2+X(2)**2-4.*X(3)
C F(3)=3.*X(1)**2-4.*X(2)+X(3)**2
C AND HERE
C
IF(METHOD.EQ. 4) FX2=F(1)
IF(METHOD.EQ. 4) RETURN
IF(IFLAG.EQ. 1) GO TO 100
FX2=C.
DO 50 I=1,IS
GX2(I)=C.
F1(I)=F(I)
50 FX2=FX2+F(I)**2
IFLAG=1
IF(METHOD.EQ. 3) RETURN
GO TO 5
100 CONTINUE
IF(METHOD.EQ. 2) GO TO 200
DO 120 I=1,IS
DF1(I)=(F(I)-F1(I))/DEL
120 GX2(ICOUNT)=GX2(ICOUNT)+2.*F1(I)*DF1(I)
121 X(ICOUNT)=X(ICOUNT)-DEL
ICOUNT=ICOUNT+1
IF(ICOUNT.LE. IS) GO TO 5
RETURN
200 DO 205 I=1,IS
DF1(I)=(F(I)-F1(I))/DEL
GX2(I)=F1(I)
205 PAR(I,ICOUNT)=DF1(I)
GO TO 121
END

```

PROGRAM LISTING B-2

METHOD=2, UNKNOWN=3, XINITL=.3, .1, .1, DEL=.005, EPS=.000001,
ITEST=25. /

PROGRAM LISTING B-3

NEWTON RAPHSON METHOD SELECTED*****

SFC	C.4939800CE C5	E6CSEC	C.2963918CE C7
DELTA X =	C.11814406E C1	X =	C.14814406E C1
DELTA X =	C.52307756E C0	X =	C.62307755E C0
DELTA X =	C.33169318E C0	X =	C.43169317E C0
DELTA X =	-0.52846862E C0	X =	C.95297194E C0
DELTA X =	-0.11960537E C0	X =	C.50347217E C0
DELTA X =	-0.58930456E-C1	X =	C.37276271E C0
DELTA X =	-0.15257811E C0	X =	C.80039382E C0
DELTA X =	-0.68232019E-C2	X =	C.49664896E C0
DELTA X =	-0.28240838E-C2	X =	C.36993862E C0
DELTA X =	-0.1500566CE-C1	X =	C.78538815E C0
DELTA X =	-0.37457305E-C4	X =	C.49661149E C0
DELTA X =	-0.1573664CE-C4	X =	C.36992288E C0
DELTA X =	-0.19058261E-C3	X =	C.78519756E C0
DELTA X =	-0.10020809E-C6	X =	C.49661139E C0
DELTA X =	-0.49638363E-C7	X =	C.36992282E C0
DELTA X =	-0.61554468E-C6	X =	C.78519693E C0
DELTA X =	C.92294171E-C8	X =	C.49661139E C0
DELTA X =	0.91793183E-C8	X =	C.36992283E C0

PROGRAM LISTING B-4

SOLUTION TO THE SYSTEM INPUT WAS OBTAINED IN 6 ITERATIONS

C.78519693E C0

C.49661139E C0

C.36592283E C0

APPENDIX B-2

FALSE-POSITION PROGRAM USER'S MANUAL

This program is designed to find the roots of the equation $F(X) = 0$ where F is a function defined through a **FUNCTION** subroutine. Moreover, the subroutine **ZEROS** is provided with a tool called recursivity which enables it to solve a set of several implicit equations such as

$$F(X, Y, Z, \dots) = 0$$

$$G(X, Y, Z, \dots) = 0$$

$$H(X, Y, Z, \dots) = 0$$

Method

Three steps are to be distinguished.

1. An iterative process in order to find a change in the sign of the function. This process may be carried out through an arithmetic or a geometric iteration.
2. When a change in the sign has been found, one of the following methods is used in order to reduce the interval in which the root lies:
 - a. if a and b define the bounds of the last interval refined by 2 above, an approximation of the graph of f on the interval (a, b) is the straight line $g(X)$ joining the points $(a, f(a))$ and $(b, f(b))$. If the evaluation of the slope of $g(X)$ is within prescribed precision, a new point c of the interval (a, b) is selected where $g(c) = 0$ and $f(c) \neq 0$ (or c is the root). The smaller new interval is (a, c) or (c, b) depending on whether $f(a), f(c) < 0$ or $f(c), f(b) < 0$. Hence this method will be elected if the slope of the straight line $g(X)$ is varying slowly and can be computed within desired accuracy.

- b. if f is rather small in the interval (a, b) and consequently a precise evaluation of the slope in "a" above is impossible, a constant weak slope is tried in an attempt to approach f on the interval (a, b) with a straight line and thus, determine a point c on (a, b) closer to the root. Hence a new interval smaller than (a, b) is determined.

The choice between those two methods is determined before each iteration through a series of tests on the results of the former iteration.

3. The output process: Before each iteration, tests will be performed in order to determine whether or not the output process has to be started. It will be started in four instances:
 - a. $F(X) = \pm$ normal zero. A solution has been found.
 - b. the interval around the root is less than or equal to the absolute error corresponding to the precision defined in the calling sequence.
 - c. there is no solution in the given interval. (See Error Messages.)
 - d. an error has been committed in the calling sequence. (See Error Messages.)

Usage

1. The subroutine ZERO is to be called as a Fortran IV subroutine through the calling sequence:
 CALL ZEROS (X, XT, FXT, RATIO, PRECI, XMIN, XMAX,
 6HTTTPRR, FONC, X1, X2, ..., X9, X10)
 or through its MAP expression.
2. Description of the arguments listed in the calling sequence:
 The argument X, FONC, X1, X2, ..., X10 must be written with identifiers. The others (except the Hollerith arguments) can be identifiers or constants according to the wish of the programmers.

- X argument of the function on which the search of the root is performed. After the return from ZEROS, the location X contains the value of the root.
- XT when ZERO is called, XT must be equal to an approximate value of the root if the programmer knows this approximate value. Otherwise select XT=0.
- FXT the corresponding value of the function. Otherwise select FXT = 0. After the return from ZEROS, XT and FXT are set to the values computed during the last step of the iterative process, so that the program is initialized for the search of a new root.
- RATIO must be chosen positive for an arithmetic iteration, greater than 1 for a geometric iteration.
- PRECI relative precision to be obtained on the variable. The maximal precision is the one of the computer, i.e., $2^{-27} = 0.74 \cdot 10^{-8}$.
- XMIN lower boundary of the interval where the search of the root is performed.
- XMAX upper boundary of the interval where the search of the root is performed.
- 6HTTTPRR this represents a code of six Hollerith characters, divided in three groups of two letters and defining:
- TT: the type of iteration
 either AR: arithmetic iteration $X_{n+1} = X_n + \text{RATIO}$
 or GE: geometric iteration $X_{n+1} = X_n * \text{RATIO}$
- PP: either the position of the initial value XT with respect to the root

IN if $XT < \text{ROOT}$
 SU if $XT > \text{ROOT}$

or the direction of variation in the interval of study

either

CR if the function is negative before the root,
 positive after,
 DC if the function is positive before the root,
 negative after,
 UN if the direction of variation is unknown. In this
 case the subroutine will manage to search the
 root in the direction along which the absolute
 value of the function is decreasing.

RR: when an exact value of the root has been found, this
 code enables the programmer to choose between the
 two values of the variable around the root, through

DF the lower value,
 EX the excess value,
 BO the best of those two values (i.e., the one for
 which the function takes its smallest absolute
 value).

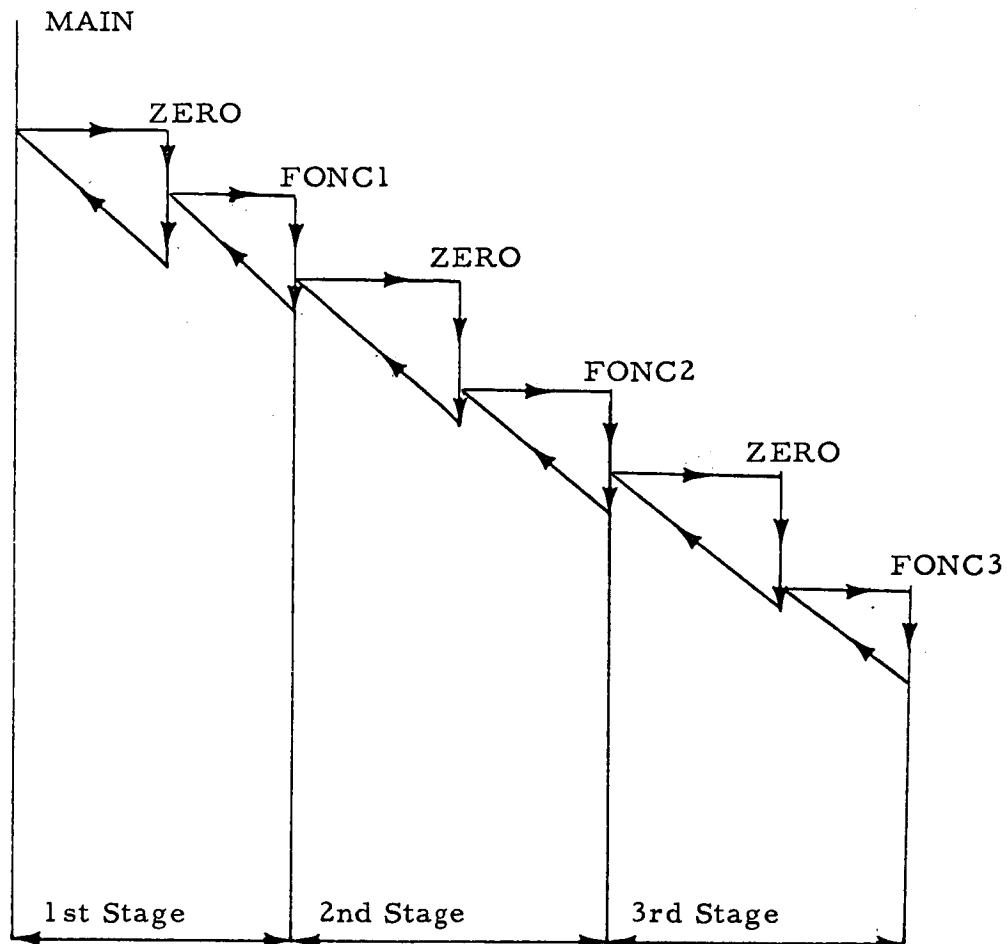
Examples 1) 6HGEINBO
 2) 6HARUNEX

FONC this is the name of the FORTRAN IV function which
 calculates the operator which is to be zeroed. This
 function has at most ten arguments; some of them
 are variables on which the search of the solution is
 performed. The others are mere parameters whose
 value is given before calling ZERO. The name of the
 function must appear within an INTERNAL instruction
 in every program, subroutine or function using ZERO.
 The function must be defined through a FUNCTION
 subroutine (see EXAMPLE).

X1, X2, X3, ..., X10 names of the arguments of FONC. They
 are FORTRAN variables (see before).

Recursivity

This tool makes it possible to call ZERO from the subroutine FUNCTION which has been called by ZERO itself.



To each stage of the computation corresponds one block of instructions and parameters, which is generated through a macro-instruction. In order to save cells, it is possible:

- 1) to set the number of blocks (or order of the recursivity) equal to the number of equations in the system to be solved.
- 2) to set the number of arguments in the functions equal to the desired number. The instruction to be modified in order to realize 1) or 2) are written under the label "parameters of dimension" (see listing):

```

      N  SET      10
      NNN SET      N + 16
      SIZE BLOCK  (1, 2, 3, ... ETMAX)

```

- 1) will be realized by setting ETMAX, in the address part of the SIZE instruction, equal to the number of equations in the system.
- 2) will be realized by setting the number of arguments in the address part of the N instruction.

Example: for a system of 2 equations with 3 arguments

```

      N  SET      3
      NNN SET      N + 16
      SIZE BLOCK  (1, 2)

```

Restrictions

For reasons of accuracy and feasibility, it is not advised to use the subroutine ZERO for systems which contain more than four equations. Beyond this number, the accuracy of the result depends a great deal on the sophistication of the equations and the run time becomes excessive.

Examples (precision desired: 0.000001)

Type of Problem	Duration (sec)	Precision Obtained
11th Order Equation	0.017	0.000001
Two Equations (of 11th order)	1.60	0.000001
Three Equations (of 11th order)	3.20	0.001

The results were obtained for highly sophisticated problems and can be considered as upper limits for each type.

Remarks

1. The subroutine can solve any system of equations as long as they are continuous with respect to each variable in the interval of study. But the programmer must pay attention to the salient features of the system he wants to solve.

Generally speaking, the subroutine ZERO will not find a solution when the four following cases occur:

- a. the system has a multiple root; this can be easily detected by the programmer by carefully studying the equations before he submits them to ZERO.
 - b. the root of the system lies inside of an interval whose length is inferior to the ratio.
 - c. one of the functions of the system starts varying in a direction which is inconsistent with the direction indicated in the Hollerith code of the calling sequence. In such a case, the root is searched in the wrong direction and the subroutine stops when it reaches one of the boundaries.
 - d. there is simply no root in the given interval.
2. If N is the mean number of iterations that will be needed to get the root of one equation, in the case of three equations, the function FONC3 will be computed N^3 times, FONC2 N^2 times and FONC1 only N times. It is then advised to choose the most sophisticated function as FONC1 and the most simple of the three as FONC3.

Error Messages

The error messages are written through an expanded WRITE 6 sequence. (This output order does not need any more than the normal

IBJOB processor). There are three messages:

1. TOO MANY ARGUMENTS
when one of the functions has more than 10 arguments.
2. INCORRECT CODING
when a mistake has been made in the Hollerith code of the calling
3. NO ROOT IN THE INTERVAL
A CALL EXIT is done after the output of the message.

This error procedure can be easily adapted to the standard one of the installation.

Storage Required

731_g, i.e., 473 cells

The whole working storage is within the subroutine itself and no common or other special storage is needed.

Sample Computer Run for the System

$$2x - 3.18309886 \quad \log \frac{1-z}{1+z} - 1 \quad = 0$$

$$\frac{\pi}{2} \times (1-y^2 z^2)(z^2-y^2)/zy^2 (1+z^2)^{-5} \quad = 0$$

$$z - .14271816 \quad = 0$$

LMSCJS
C54

- EFN SOURCE STATEMENT - IFN(S) -

```

EXTERNAL FØNC1
XT=0.01
FXT=0.
CALL ZERØS(X,XT,FXT,1.2,0.000001,-5.,5.,6HGEINBØ,FØNC1,X,Y,Z)
WRITE(6,100) X,Y,Z
100 FØRMAT(1HØ,1ØX,2HX=,E15.8,5X,2HY=,E15.8,5X,2HZ=,E15.8 )
STØP
END

```

LMSCJS
C54A

- EFN SOURCE STATEMENT - IFN(S) -

```

FUNCTION FØNC1(X,Y,Z)
EXTERNAL FØNC2
YT=0.01
FYT=0.
CALL ZERØS(Y,YT,FYT,0.01,0.000001,-2.,5.,6HARINBØ,FØNC2,X,Y,Z)
FØNC1=2.*X-3.18309886*ALØG((1.-Z)/(1.+Z))-1.
RETURN
END

```

LMSCJS
C54B

- EFN SOURCE STATEMENT - IFN(S) -

```

FUNCTION FØNC2(X,Y,Z)
EXTERNAL FØNC3
ZT=0.1
FZT=0.
CALL ZERØS(Z,ZT,FZT,0.01,0.000001,-2.,5.,6HARINBØ,FØNC3,X,Y,Z )
FØNC2=1.57079632*X*(1.-Y**2*Z**2)*(Z**2-Y**2)
1 /Z/Y**2/(1.+Z**2)-5.
RETURN
END

```

LMSCJS
C54C

- EFN SOURCE STATEMENT - IFN(S) -

FUNCTION F0NC3(X,Y,Z)
F0NC3=Z-0.14271816
RETURN
END

LMSCJS IBLDR

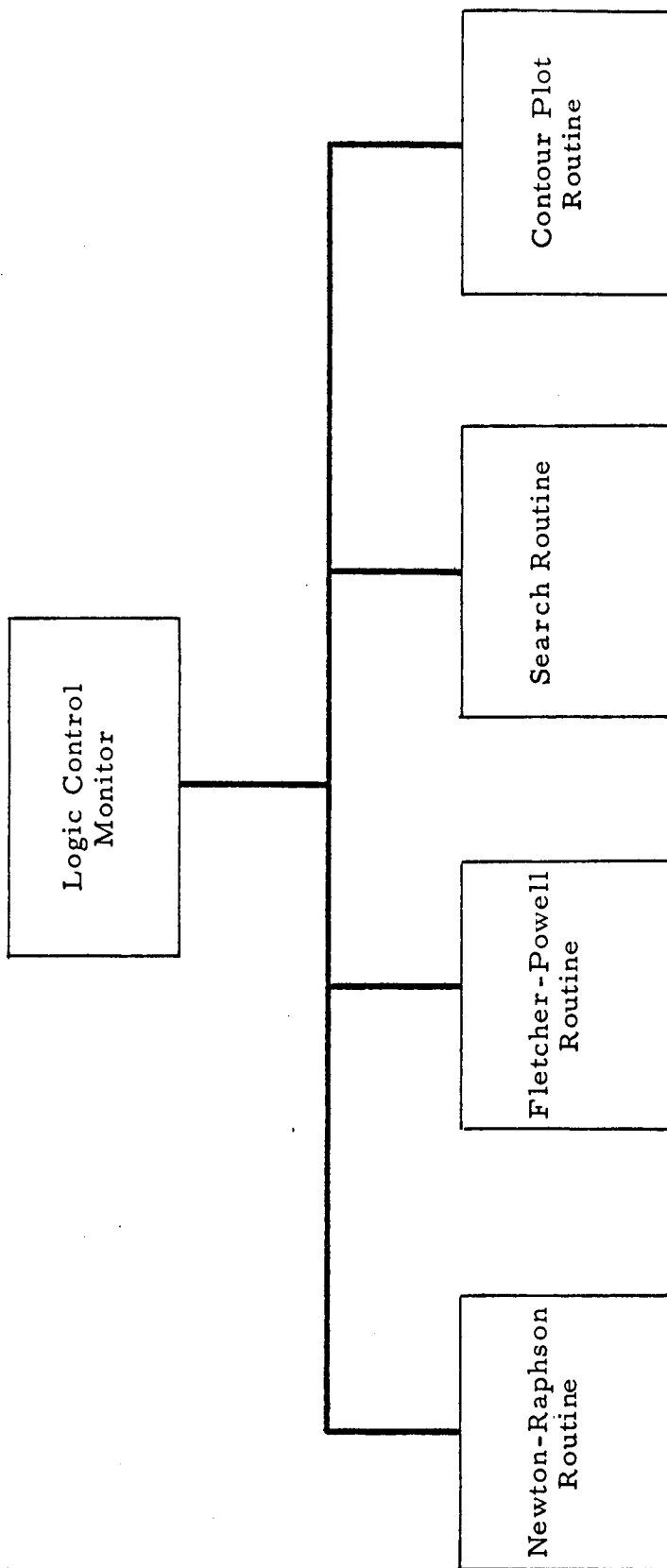
UNUSED CORE

77775 THRU 77777

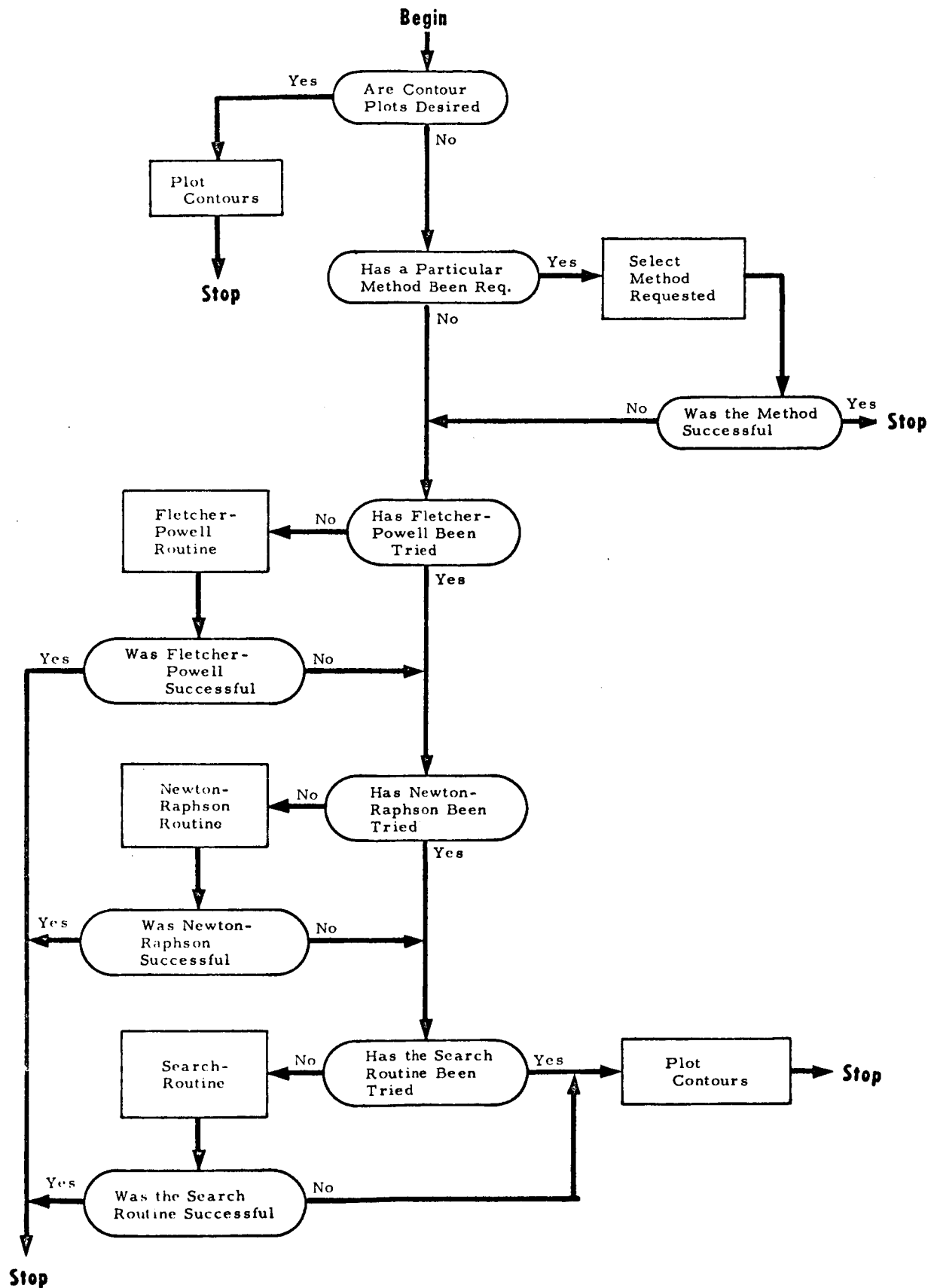
X= 0.42591367E-01 Y= 0.41400141E-01 Z= 0.14271816E 00

APPENDIX C-1
PROGRAMMER'S MANUAL

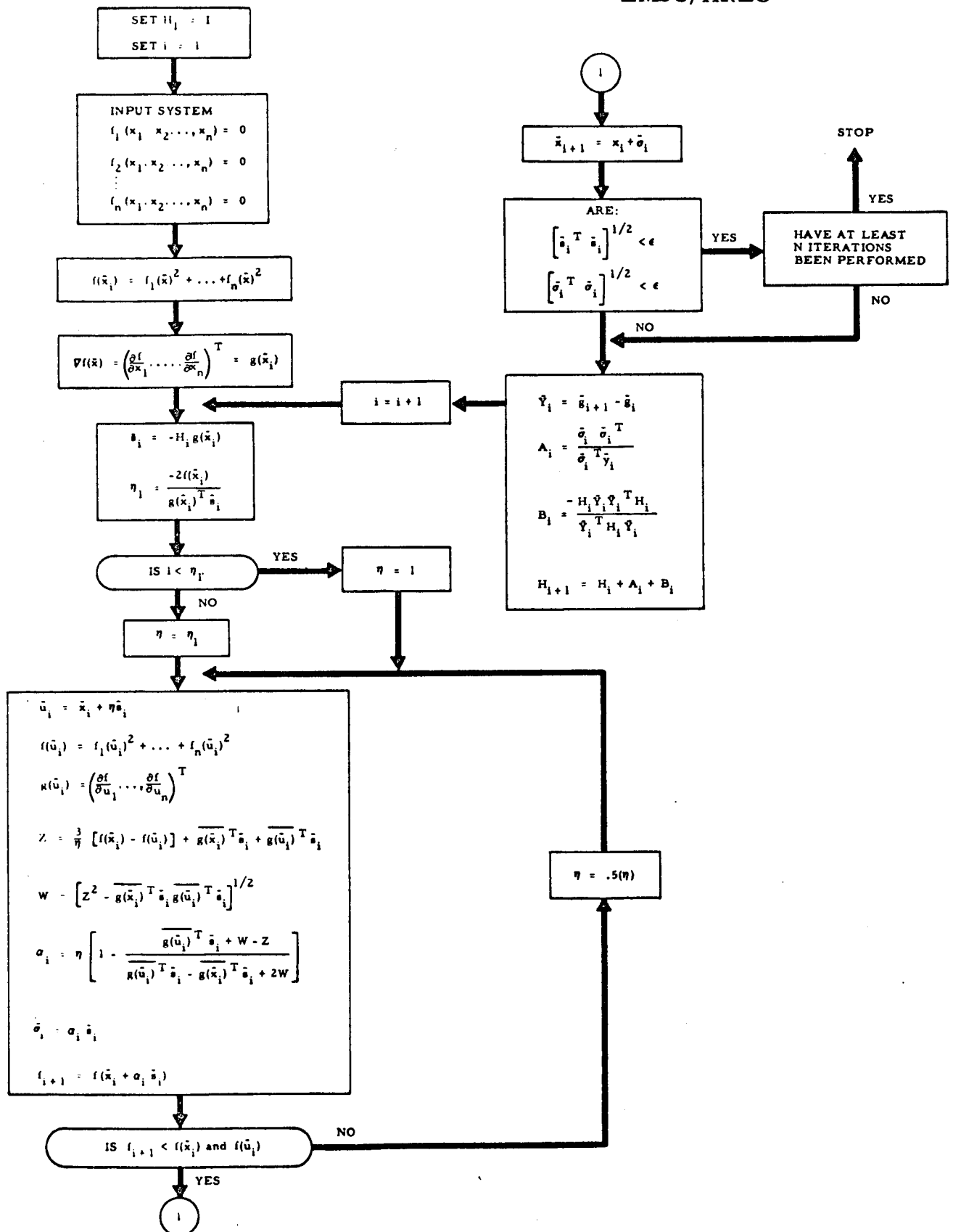
A description of each subroutine with flow charts is given in this section and is intended for use by a programmer attempting to understand or change the computer program. A list of symbols is given with R indicating REAL and I indicating INTEGER variables. A listing of the program is found at the end of this section.



Composite Program



Logic Flow of Composite Program



Flow Chart of Fletcher-Powell Technique

LIST OF SYMBOLS

By Common Blocks

CNTR	XL	R	= left most position for x-axis of contour plots
	XR	R	= right most position for x-axis of contour plots
	YB	R	= bottom most position of y-axis of contour plots
	YT	R	= top most position for y-axis of contour plots
	N	I	= number of contour plots to be plotted
DATA	F(1300)	R	= table of function values at each grid line intersection on contour plot frame
	X(1300)	R	= table of x-values at each grid line intersection on contour plot frame
	Y(1300)	R	= table of y-values at each grid line intersection on contour plot frame
ERROR	ITEST	I	= maximum number of iterations to be allowed before solution technique considered unsatisfactory
	IER	I	= error flag $\neq 0$ indicates solution technique has failed
H1	H(20, 20)	R	= identity matrix initially and modified in search routine in computations of terms dependent on gradient leading to minimum
INIT	IS	I	= size of system to be solved (number of unknowns)
	IN	I	= order of the system to be solved
	DEL	R	= increment for rate of change of unknowns in computing partials
	EPS	R	= test criterion indicating accuracy desired in solution
	METHOD	I	= flag indicating solution technique to be used

	ITER	I	= counter, keeps a running count of number of iterations used in solution techniques
SRCH	P(21, 20)	R	= table of simplex point for search routine
	ALPHA	R	= reflection coefficient in search routine
	BETA	R	= contraction coefficient in search routine
	GAMMA	R	= expansion coefficient in search routine

By Subroutines

DRIVER	COMMON	ERROR	
		H1	
		INIT	
	ITABLE(3)	I	= table of solution techniques attempted that have failed
	K1	I	= counter, counts number of solution techniques tried
	X(20)	R	= table of solutions to unknowns. Initially contains initial estimates to roots
CONTOR	COMMON	CNTR	
		DATA	
		INIT	
	DF	R	= contour value increment for successive contour plots
	DX	R	= x increment for construction of grid lines or contour plot
	DY	R	= y increment for construction of grid lines on contour plot
	FC(100)	R	= table of values of contours to be plotted
	FMAX	R	= maximum functional value on contour grid
	FMIN	R	= minimum functional value on contour grid
	F0	R	= contour functional value for which x and y are plotted
	F1	R	= upper functional value available closest to F0
	F2(20)	R	= lower functional value available closest to F0 (only F2(1) is used here)

	IBOX	I	= grid box number (there are 1225 in total grid frames) under consideration for contour plot values
	IBX	I	= column number (x-grid line) under consideration
	IBY	I	= row number (y-grid line) under consideration
	IFLAG	I	= first time through flag (if 0 first time)
	IFL2	I	= even (= 0) and odd (= 1) row under consideration
	ITABLE(1250)	I	= table of flags determining direction of box diagonal (= 0 implies N to N + 3 and = 1 implies N + 1 to N + 2)
	IX	I	= a particular value of ITABLE
	I1	I	= index for first point
	I2	I	= index for second point
	J	I	= index used in loading ITABLE
	NS	I	= plotting symbol selector
	V	R	= number of contours - 1
	V1	R	= functional value at first point
	V2	R	= functional value at second point
	X1	R	= x-value to be plotted
	X2	R	= y-value to be plotted
	XV	R	= intermediate storage for most recent x-value in computing contour values
	YV	R	= intermediate storage for most recent y-value in computing contour values
EVAL	COMMON	INIT	
	DFL(20)	R	= table of increments of functions for computing partials
	F(20)	R	= table of functional values of system of equations
	FX2	R	= sum of the squares of the functions
	F1(20)	R	= functional values
	GX2(20)	R	= functional values
	ICOUNT	I	= counter for number of times through routine computing partials

IFLAG	I	= first time through flag (= 0 first time)
PAR(20, 20)	R	= table of partial derivatives
X(20)	R	= table of x-values for system
FPOWEL	COMMON	ERROR
		H1
		INIT
A(20, 20)	R	= array of storage for A_i matrix
ALPHA	R	= storage for α in minimization process = $\eta(1 - (T2 + W - Z)/(T2 - T1 + 2W))$
ARG	R	= square root argument for $Z^2 = T1 - T2$
A1	R	= storage for $\sigma_i^T Y_i$
A2	R	= storage for $Y_i^T H_i Y_i$
B(20, 20)	R	= storage for B_i matrix = $\frac{-H_i \bar{Y}_i \bar{Y}_i^T H_i}{\bar{Y}_i^T H_i \bar{Y}_i}$
D1	R	= storage for $g(X_i)^T \bar{S}_i$
ETA	R	= storage for η
ETA1	R	= storage for $\eta_1 = (-2F)/D1$
E1(20, 20)	R	= $-H_i \bar{Y}_i$
E2(20, 20)	R	= $E1 \bar{Y}_i^T$
F	R	= value of sum of squares of functions
FXAS	R	= value of sum of squares of functions at new iterative point
G(20)	R	= gradient of functions of X_i
GU(20)	R	= gradient of functions of U
GXAS(20)	R	= gradient of functions of X_{i+1}
IMIN	I	= maximum number of times through minimization process allowable
S(20)	R	= $-H_i g(\bar{X}_i)$
SIG(20)	R	= x increment in iterative procedure
T1	R	= $[\bar{S}_i^T S_i]^{1/2}$

	T2	R	$= \left[\bar{\sigma}_i^T \bar{\sigma}_i \right]^{\frac{1}{2}}$
	U(20)	R	$= \bar{X}_i + \eta \bar{S}_i$
	w	R	$= \sqrt{Z^2 - T1 \ T2}$
	W(20, 20)	R	= dummy storage for calling EVAL
	X(20)	R	= x-iterates at t_i
	XAS(20)	R	= x-iterates at t_{i+1}
	Y(20)	R	$= GXAS_i - G_i$
	Z	R	$= 3/\eta (F - FU) + T1 + T2$
INITIAL	COMMON	CNTR	
		ERROR	
		H1	
		INIT	
		SRCH	
	IERR	I	= error flag for MAVRICK routine ($\neq 0$ indicated error)
	X(20)	R	= initial estimates to roots
	X1(20)	R	= save storage of initial estimates
LINTBL	COMMON	DATA	
	F1	R	= functional value at first point
	F2(20)	R	= functional value at second point (only F2(1) is used)
	I	I	= row index
	J	I	= column index
	X1	R	= interpolated x-value
	Y1	R	= interpolated y-value
NEWRAP	COMMON	ERROR	
		INIT	
	DX(20, 1)	R	= x increment for x-iterates
	F(20)	R	= values of functions
	PAR(20, 20)	R	= partial derivatives
	WK(20, 20)	R	= inverse matrix (D^{-1})

	X(20)	R	= x-values
PRINT1	COMMON	INIT	
	X(20)	R	= roots to system of equations
SRCH1	COMMON	INIT	
		SRCH	
	G1(20)	R	= functional values
	ITABLE(2)	I	= keeps track of location of YH(ITABLE(1)) and YL(ITABLE(2))
	IX1	I	= index for YH
	IX2	I	= index for YL
	N	I	= size of system + one
	PCENT(20)	R	= centroid
	PH(20)	R	= points yielding highest functional value
	PL(20)	R	= points yielding lowest functional value
	PNTS	R	= system size (floating point)
	PS(20)	R	= $P^* = (1 + \alpha) \bar{P} - \alpha Ph$
	PSS(20)	R	= $P^{**} = (1 + \gamma) P^* - \gamma \bar{P}$
	W1(20, 20)	R	= dummy storage for calling EVAL
	X(20)	R	= roots to equations
	X1(20)	R	= x-values input to EVAL
	Y(21)	R	= y-values = $F(p)$
	YH	R	= highest y
	YL	R	= lowest \bar{Y}
	YS	R	= $y^* = y(P^*)$
	YSS	R	= $y^{**} = y(P^{**})$

SUBROUTINE DRIVER

The driver controls entry to all solution techniques including entry to the input and output routines.

Equations Used: None

Labeled Common: ERROR

H1

INIT

Dimensioned Storage: ITABLE (3)

X(20)

Called from: System

External References: CONTOR

FPOWELL (X)

INITAL (X)

NEWRAP (X)

PRINT1 (X)

SRCH1 (X)

Input: None (control region)

Output: None (control region)

SUBROUTINE INITAL (X)

Reads input data, checks for read errors, and initializes estimates to roots.

Equations Used: None

Labeled Common: CNTR
 ERROR
 H1
 INIT
 SRCH

Dimensioned Storage: X (20)
 X1 (20)

Called from: DRIVER

External References: MAVRIK

Input: Data cards

Output: Initial estimates to roots, desired computation options and constants

SUBROUTINE EVAL (X, FX2, GX2, PAR)

Evaluates the system of equations for a given set of x values, computes sum of squares of functions and partial derivatives.

Equations Used:

$$\begin{bmatrix} \frac{\partial f_1}{\partial x_1}, \frac{\partial f_1}{\partial x_2}, \dots, \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1}, \frac{\partial f_2}{\partial x_2}, \dots, \frac{\partial f_2}{\partial x_n} \\ \vdots \\ \frac{\partial f_n}{\partial x_1}, \frac{\partial f_n}{\partial x_2}, \dots, \frac{\partial f_n}{\partial x_n} \end{bmatrix}$$

where

$$\frac{\partial f_i}{\partial x_i} = \sum_{k=1}^n 2f_k \Delta f_k$$

with

$$\Delta f_k = (f_k - f_{1k})/\Delta x_i$$

$$f_k = f(x_k) \text{ for } i \neq n$$

$$f_k = f(x_k + \Delta x) \text{ for } i = k$$

Labeled Common: INIT

Dimensioned Storage: DF1 (20)
DX (20)
F (20)
F1 (20)
GX2 (20)
PAR (20, 20)
X (20)

Called from: CONTOR
FPOWEL (X)
NEWRAP (X)
SRCH1 (X)

External References: None

Input: X-iterates

Output: Matrix of partials, sum of squares of functions, and values of each
function in system

SUBROUTINE FPOWEL (X)

The Fletcher-Powell iterative technique is for solving simultaneous non-linear algebraic equations.

Equations Used:

$$f(\bar{x}_i) = f_1(\bar{x})^2 + \dots + f_n(\bar{x})^2$$

$$\nabla f(\bar{x}) = \left(\frac{\partial f}{\partial x_1} \dots \frac{\partial f}{\partial x_n} \right)^T = g(\bar{x}_i)$$

$$\bar{S}_i = -H_i g(\bar{x}_i)$$

$$\eta_1 = \frac{-2f(\bar{x}_i)}{g(\bar{x}_i)^T \bar{S}_i}$$

$$\bar{u}_i = \bar{x}_i + \eta \bar{S}_i$$

$$f(\bar{u}_i) = f_1(\bar{u}_i)^2 + \dots + f_n(\bar{u}_i)^2$$

$$u(\bar{u}_i) = \left(\frac{\partial f}{\partial u_1} \dots \frac{\partial f}{\partial u_n} \right)^T$$

$$Z = \frac{3}{\eta} \left[f(\bar{x}_i) - f(\bar{u}_i) \right] + \overline{g(\bar{x}_i)}^T \bar{S}_i + \overline{g(\bar{u}_i)}^T \bar{S}_i$$

$$W = \left[Z^2 - \overline{g(\bar{x}_i)}^T \bar{S}_i \overline{g(\bar{u}_i)}^T \bar{S}_i \right]^{\frac{1}{2}}$$

$$\alpha_i = \eta \left[1 - \frac{\overline{g(\bar{u}_i)}^T \bar{S}_i + W - Z}{\overline{g(\bar{u}_i)}^T S_i - \overline{g(\bar{x}_i)}^T \bar{S}_i + 2W} \right]$$

$$\sigma_i = \alpha_i \bar{S}_i$$

$$f_{i+1} = f(\bar{x}_i + \alpha_i \bar{S}_i)$$

$$\bar{x}_{i+1} = \bar{x}_i + \bar{\sigma}_i$$

$$t_1 = \left[\bar{S}_i^T \bar{S}_i \right]^{\frac{1}{2}}$$

$$t_2 = \left[\bar{\sigma}_i^T \bar{\sigma}_i \right]^{\frac{1}{2}}$$

$$\bar{Y}_i = \bar{g}_{i+1} - \bar{g}_i$$

$$A_i = \frac{\bar{\sigma}_i \bar{\sigma}_i^T}{\bar{\sigma}_i^T \bar{Y}_i}$$

$$B_i = \frac{-H_i \bar{Y}_i \bar{Y}_i^T H_i}{\bar{Y}_i^T H_i \bar{Y}_i}$$

$$H_{i+1} = H_i + A_i + B_i$$

Labeled Common: ERROR

H1

INIT

Dimensioned Storage: A (20, 20)
 B (20, 20)
 E1 (20, 20)
 E2 (20, 20)
 G (20)
 G4 (20)
 GXAS (20)
 S (20)
 SIG (20)
 U (20)
 W1 (20, 20)
 X (20)
 XAS (20)
 Y (20)

Called from: DRIVER

External References: EVAL (X, F, G, W1)
 EVAL (U, FU, GU, W1)
 EVAL (XAS, FXAS, GXAS, W1)

Input: Initial estimates (x_i)

Output: Roots to system of equations (x_n)

SUBROUTINE NEWRAP (X)

The Newton-Raphson iterative technique for solving simultaneous non-linear algebraic equations.

Equations Used:

$$\left\{ \begin{array}{l} F_{x_1}^1 + F_{x_2}^2 + \dots + F_{x_n}^n = -f_1(x_1, x_2, \dots, x_n) \\ F_{x_1}^2 + F_{x_2}^2 + \dots + F_{x_n}^2 = -f_2(x_1, x_2, \dots, x_n) \\ F_{x_1}^n + F_{x_2}^n + \dots + F_{x_n}^n = -f_n(x_1, x_2, \dots, x_n) \end{array} \right.$$

where

$$F_{x_1}^1 = \left. \frac{\partial f_1}{\partial x_1} \right|_{(i-1)} \Delta x_1, \quad F_{x_1}^2 = \left. \frac{\partial f_2}{\partial x_1} \right|_{(i-1)} \Delta x_1, \quad \text{etc.}$$

$$AX_{\delta} = B$$

where

$$A = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \dots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \dots & \dots & \frac{\partial f_n}{\partial x_n} \\ \vdots & & & \\ \frac{\partial f_n}{\partial x_1} & \dots & \dots & \frac{\partial f_n}{\partial x_n} \end{bmatrix}$$

$$X_{\delta} = \begin{bmatrix} \Delta x_1 \\ \Delta x_2 \\ \vdots \\ \Delta x_n \end{bmatrix}$$

$$B = \begin{bmatrix} -f_1(x_1, x_2, \dots, x_n) \\ \vdots \\ -f_n(x_1, x_2, \dots, x_n) \end{bmatrix}$$

$$X_i = X_i + X_{\delta_i}$$

Labeled Common: ERROR

INIT

Dimensioned Storage: DX(20, 1)

F (20)

PAR (20, 20)

WK (20, 20)

X (20)

Called from: DRIVER

External References: EVAL (X, FX2, F, PAR)

GASSIM (WK, IS, 1, DET, DX)

Input: Initial estimates (x_i)

Output: Roots to system of equations (x_n)

SUBROUTINE SRCH1 (X)

The Simplex Method for function minimization of Nalder and Mead for locating minima.

Equations Used:

$$F = \sum_{i=1}^n (f_i)^2$$

$$Y = F(P)$$

$$Y_h = F(P_h)$$

$$Y_l = F(P_l)$$

$$\bar{P} = \frac{1}{n} \sum_{\substack{i=1 \\ i \neq h}}^n P_i$$

$$P^* = (1 + \alpha) \bar{P} - \alpha P_h$$

$$Y(P^*) = Y^*$$

$$P^{**} = (1 + \gamma) P^* - \bar{P}$$

$$Y^{**} = Y(P^{**})$$

$$P^{**} = \beta P_h + (1 - \beta) \bar{P}$$

$$P_i = \frac{1}{n} \sum_{l=1}^n (P_i + P_l)$$

Labeled Common: ERROR

INIT

SRCH

Dimensioned Storage: G1 (20)

PCENT (20)

PH (20)

PL (20)

PS (20)

PSS (20)

W1 (20, 20)

X (20)

X1 (20)

Y (21)

Called from: DRIVER

External References: EVAL (X1, Y(I), G1, W1)

EVAL (X1, YS, G1, W1)

EVAL (X1, YSS, G1, W1)

Input: Simplex points

Output: Roots to system of equations

SUBROUTINE PRINT1 (X)

Prints the solution to the system of equations input.

Equations Used: None

Labeled Common: INIT

Dimensioned Storage: X (20)

Called from: DRIVER

External References: None

Input: Solutions to system

Output: Printed solutions

SUBROUTINE CONTOR

Plots contours on a grid of 35 lines by 35 lines.

Equations Used:

$$\frac{F_{ij \text{ max}} - F_{ij \text{ min}}}{N - 1} = \Delta F$$

$$F_i = F_{ij \text{ min}} + (i - 1) \Delta F$$

or
$$F_i = F_{ij \text{ min}} + (SF)^{i-1} \Delta F$$

Labeled Common: CNTR

DATA

INIT

Dimensioned Storage: FC (100)

F2 (20)

PAR (20, 20)

X1 (20)

Called from: DRIVER

External References: CAMRAV (9)

GRIDIV (1, XL, XR, YB, YT, DX, DY, -6, -6, -6, -6,
-3, -3)

LINTBL (I1, I2, X1, Y1, F1, F22, F0)

POINTV (X1, Y1, -NS)

Input: Grid limits and number of contours desired

Output: Contour plots

NONLINEAR EQUATIONS

```

$IBFTC C69      DECK
COMMON /INIT/  IS,IN,DEL,EPS,METHOD,ITER
COMMON /H1/   H(20,20)
COMMON /ERROR/ ITEST,IER
DIMENSION ITABLE(3)
DIMENSION X(20)
ITEST=50
DEL=.001
EPS=.00001
5 K1=0
DO 6 I=1,3
6 ITABLE(I)=0
IER=0
7 CALL INITIAL(X)
ITER=0
IF(METHOD.EQ. 1) GO TO 100
IF(METHOD.EQ. 2) GO TO 200
IF(METHOD.EQ. 3) GO TO 300
IF(METHOD.EQ. 4) GO TO 400
GO TO 500
100 CALL FPOWEL(X)
K1=K1+1
IF(IER.NE. 0) GO TO 150
CALL PRINT1(X)
GO TO 5
150 ITABLE(K1)=IER
WRITE(6,151) ITER
151 FORMAT(38H1FLETCHER POWELL UNSATISFACTORY AFTER ,I5,11H ITERATIONS
1
,
CALL PRINT1(X)
GO TO 600
200 CALL NEWRAP(X)
K1=K1+1
IF(IER.NE. 0) GO TO 250
CALL PRINT1(X)
GO TO 5

```

```

250 ITABLE(K1)=IER
WRITE(6,251) ITER
251 FORMAT(37H1NEWTON RAPHSON UNSATISFACTORY AFTER ,I5.11H ITERATIONS)
GO TO 600
300 CALL SRCH1(X)
K1=K1+1
IF(IER.NE.0) GO TO 350
CALL PRINT1(X)
GO TO 5
350 ITABLE(K1)=IER
WRITE(6,351)
351 FORMAT(36H1SEARCH METHOD UNSATISFACTORY AFTER ,I5.11H ITERATIONS)
GO TO 600
400 CALL CONTOR
GO TO 5
500 WRITE(6,501)
501 FORMAT(29H0ALL ROUTINES HAVE BEEN TRIED)
STOP
600 CONTINUE
J1=1
605 DO 608 I=1,3
DO 608 J=1,3
IF(J1.EQ. ITABLE(J)) J1=J1+1
608 CONTINUE
METHOD = J1
GO TO 7
END

```

```

$IBFTC C69A  DECK
SUBROUTINE INITIAL(X)
COMMON /CNTR/ XL,XR,YB,YT,N,SF
COMMON /H1/ H(20,20)
COMMON /INIT/ IS,IN,DEL, EPS , METHOD,ITER
COMMON /SRCH/ P(21,20),ALPHA,BETA,GAMMA
COMMON /ERROR/ ITEST,IER
DIMENSION X(20),X1(20)
SF = 1.
IF( IER .NE. 0) GO TO 100
CALL MAVRIK(IERR,6HUKNOWN,IS,SHORDER,IN,6HXINITL,X,3HDEL,DEL,
1 3HEPS,EPS,6HMETHOD,METHOD,6HPOINTS,P,5HALPHA,ALPHA,4HBETA,BETA,
2 5HGAMMA,GAMMA,5HITEST,ITEST,2HXL,XL,2HXR,XR,2HYB,YB,2HYT,YT,
3 1HN,N,2HSF,SF)
IF( IERR .EQ. 0) GO TO 10
WRITE(6,5)
5 FORMAT(1H1,15HERROR IN MAVRIK)
STOP
10 DO 20 I=1,IS
X1(I)=X(I)
DO 20 J=1,IS
H(I,J)=0.
IF(I.EQ.J)H(I,J)=1.
20 CONTINUE
RETURN
100 DO 101 I=1,IS
101 X(I)=X1(I)
RETURN
END

```

```

$1BFTC C69B      DECK
SUBROUTINE EVAL(X,FX2,GX2,PAR)
COMMON /INIT/ IS,IN,DEL, EPS , METHOD,ITER
DIMENSION X(20),F(20),GX2(20) ,PAR(20,20),DX(20)
DIMENSION F1(20), DF1(20)
IF(METHOD .EQ. 4) GO TO 10
ICOUNT=1
IFLAG=0
GO TO 10
5 CONTINUE
  X(ICOUNT)=X(ICOUNT)+DEL
  10 CONTINUE
C WRITE IN SYSTEM BEGINNING IN COLUMN 7 WITH VARIABLES AS X(I) SUCH AS
C   F(1)=X(1)**2+X(2)-11.
C   F(2)=X(2)**2+X(1)-7.
C FOR THE SYSTEM
C   X**2+Y-11.=0.
C   Y**2+X- 7.=0.
C WHERE
C   X=X(1)
C   Y=X(2)
C
C PLACE SYSTEM BETWEEN HERE
  F(1)=(X(1)**2-1.)*2+(X(1)**2-X(2)-1.-EPS)**2
C AND HERE
C
  IF(METHOD .EQ. 4) FX2=F(1)
  IF(METHOD .EQ. 4) RETURN
  IF(IFLAG .EQ. 1) GO TO 100
  FX2=0.
  DO 50 I=1,IS
    GX2(I)=0.
    F1(I)=F(I)
    50 FX2=FX2+F(I)**2
    IFLAG=1
    IF(METHOD .EQ. 3) RETURN
    GO TO 5
    100 CONTINUE
    IF(METHOD .EQ. 2) GO TO 200
    DO 120 I=1,IS

```



```

DF1(I)=(F(I)-F1(I))/DEL
120 GX2(ICOUNT)=GX2(ICOUNT)+2.*F1(I)*DF1(I)
121 X(ICOUNT)=X(ICOUNT)-DEL
    ICOUNT=ICOUNT+1
    IF(ICOUNT.LE. IS) GO TO 5
    RETURN
200 DO 205 I=1,IS
    DF1(I)=(F(I)-F1(I))/DEL
    GX2(I)=F1(I)
205 PAR(I,ICOUNT)=DF1(I)
    GO TO 121
END
C69B0400
C69B0410
C69B0420
C69B0430
C69B0440
C69B0450
C69B0460
C69B0470
C69B0480
C69B0490
C69B0500
C69B0510

```

```

$IBFTC C69C  DECK
SUBROUTINE FPOWER(X)
  DIMENSION G(20),GU(20),U(20),X(20),Y(20),XAS(20),GXAS(20),
  1  S(20), SIG(20),  A(20,20), B(20,20)
  DIMENSION EI(20,20), E2(20,20)
  COMMON /INIT/ IS,IN,DEL, EPS , METHOD,ITER
  COMMON /H1/ H(20,20)
  COMMON /ERROR/ ITEST,IER
  DIMENSION W1(20,20)
  WRITE(6,930)
930  FORMAT(42H1FLETCHER POWELL METHOD SELECTED*****
  CALL SCLOCK( DATE, TIME, ESEC, E60SEC)
  WRITE(6,931) ESEC, E60SEC
931  FORMAT(6H0SEC ,E20.8,10H E60SEC ,E20.8)
  CALL EVAL(X,F,G,W1)
  5  ITER=ITER+1
  IF(ITER.GT. ITEST) GO TO 510
  DO 10 I=1,IS
    S(I)=0.
    DO 10 J=1,IS
      DO 10 S(I)=S(I)-H(I,J)*G(J)
      DI=0.
      DO 15 I=1,IS
        DI=DI+G(I)*S(I)
      ETA1=(-2.*F)/DI
      ETA=ETA1
      IF(1. .LT. ETA1) ETA=1.
      IMIN=0
19  CONTINUE
      IMIN=IMIN+1
      DO 20 I=1,IS
        U(I)=X(I)+ETA*S(I)
        CALL EVAL(U,FU,GU,W1)
        T1=D1
        T2=0.
        DO 30 I=1,IS
          T2=T2+GU(I)*S(I)
          Z=3./ETA*(F-FU)+T1+T2
          ARG=Z**2-T1*T2
          IF( ARG.GT. 0. ) GO TO 34

```

```

ETA=2.*ETA
GO TO 19
34 W=SQRT(ARG)
ALPHA=ETA*(1.-((T2+W-Z)/(T2-T1+2.*W)))
DO 35 I=1,IS
35 SIG(I)=ALPHA*S(I)
DO 40 I=1,IS
40 XAS(I)=X(I)+SIG(I)
CALL EVAL(XAS,FXAS,GXAS,W1)
WRITE(6,43) ITER,FXAS
43 FORMAT(35HOSUM OF SQUARES OF FUNCTIONS AFTER ,15,15H ITERATIONS =
1 ,E20.8)
IF(FXAS .LT. F .AND. FXAS .LT. FU) GO TO 45
ETA=ETA*.75
IF(IMIN .GT. 25) GO TO 520
GO TO 19
45 CONTINUE
T1=0.
T2=0.
DO 50 I=1,IS
T1=T1+S(I)*S(I)
50 T2=T2+SIG(I)*SIG(I)
T1=SQRT(T1)
T2=SQRT(T2)
IF(T1 .LT. EPS .AND. T2 .LT. EPS .AND. ITER .GE. IN) GO TO 500
DO 60 I=1,IS
60 Y(I)=GXAS(I)-G(I)
DO 57 I=1,IS
X(I)=XAS(I)
57 G(I)=GXAS(I)
F=FXAS
A1=0.
A2=0.
DO 62 I=1,IS
62 A1=A1+SIG(I)*Y(I)
DO 65 I=1,IS
DO 65 J=1,IS
A(I,J)=(SIG(I)*SIG(J))/A1
65 A2=Y(J)*H(I,J)*Y(I)+A2
DO 66 I=1,IS

```

```

E1(I,1)=0.
DO 66 IJ=1,IS
66 E1(I,1)=E1(I,1)+(-H(I,IJ)*Y(IJ))
DO 67 I=1,IS
DO 67 J=1,IS
67 E2(I,J)=E1(I,1)*Y(J)
DO 68 I=1,IS
DO 68 J=1,IS
B(I,J) = 0.
DO 68 IJ=1,IS
68 B(I,J)=B(I,J)+E2(I,IJ)*H(IJ,J)
DO 70 I=1,IS
DO 70 J=1,IS
B(I,J)=B(I,J)/A2
70 H(I,J)=H(I,J)+A(I,J)+B(I,J)
GO TO 5
500 IER=0
RETURN
510 IER=1
RETURN
520 WRITE(6,521) IMIN
521 FORMAT(42H MINIMIZATION ATTEMPTS IN ONE ITERATION = ,I3)
IER=1
RETURN
END

```

```

$IBFTC C69D    DECK
SUBROUTINE NEWRAP(X)
COMMON /INIT/ IS,IN,DEL, EPS , METHOD,ITER
COMMON /ERROR/ ITEST,IER
DIMENSION X(20),F(20),PAR(20,20),WK(20,20),DX(20,1),FB(20,1)
WRITE(6,930)
930 FORMAT(41H)NEWTON RAPHSON METHOD SELECTED*****
CALL SCLOCK( DATE, TIME, ESEC, E60SEC)
WRITE(6,931) ESEC, E60SEC
931 FORMAT(6H)SEC ,E20.8,10H E60SEC ,E20.8)
7 CONTINUE
CALL EVAL(X,FX2,F,PAR)
DO 6 I=1,IS
6 DX(I,1)=-F(I)
DO 5 I=1,IS
DO 5 J=1,IS
5 WK(I,J)=PAR(I,J)
CALL GASSIM(WK,IS,1,DET,DX)
DO 20 K1=1,IS
X(K1)=X(K1)+DX(K1,1)
WRITE(6,101) DX(K1,1),X(K1)
101 FORMAT(10H)DELTA X = ,E20.8,6H X = ,E20.8)
20 CONTINUE
50 CONTINUE
ITER=ITER+1
IF(ITER .GT. ITEST) GO TO 500
DO 55 I=1,IS
IF(ABS(DX(I,1)).GT. EPS) GO TO 7
55 CONTINUE
IER=0
RETURN
500 IER=2
RETURN
END

```

```

$1BFTC C69E DECK
SUBROUTINE SRCH1(X)
COMMON /INIT/ IS,IN,DEL,EPS,METHOD,ITER
COMMON /SRCH/ P(21,20),ALPHA,BETA,GAMMA
COMMON /ERROR/ ITEST,IER
DIMENSION Y(21),ITABLE(2),X1(20),G1(20),PS(20),PSS(20),
1 PH(20),PL(20),PCENT(20)
DIMENSION W1(20,20),X(20)
WRITE(6,630)
630 FORMAT(33H1SEARCH METHOD SELECTED*****
CALL SCLOCK(DATE,TIME,ESEC,E60SEC)
WRITE(6,931) ESEC,E60SEC
931 FORMAT(6H0SEC ,E20.8,10H E60SEC ,E20.8)
N=IS+1
PNTS=IS
5 DO 10 I=1,N
DO 6 J=1,IS
6 X1(J)=P(I,J)
ITER=ITER+1
IF(ITER.GT. ITEST) GO TO 900
CALL EVAL(X1,Y(I),G1,W1)
10 CONTINUE
YH=Y(I)
YL=Y(I)
ITABLE(1)=1
ITABLE(2)=1
DO 14 I=2,N
IF(Y(I).GT. YH) GO TO 12
IF(Y(I).LT. YL) GO TO 13
GO TO 14
12 YH=Y(I)
ITABLE(1)=I
GO TO 14
13 YL=Y(I)
ITABLE(2)=I
14 CONTINUE
IX1=ITABLE(1)
IX2=ITABLE(2)
DO 20 I=1,IS
PCENT(I)=0.

```

```

PH(I)=P(IX1,I)
20 PL(I)=P(IX2,I)
  IF(YL.LT.EPS) GO TO 200
C  COMPUTE CENTROID
  DO 25 I=1,N
    IF(IX1.EQ.I) GO TO 25
  DO 23 J=1,IS
    PCENT(J)=PCENT(J)+P(I,J)
  25 CONTINUE
  DO 26 I=1,IS
    PCENT(I)=PCENT(I)/PNPTS
  DO 30 I=1,IS
    PS(I)=(1.+ALPHA)*PCENT(I)-ALPHA*PH(I)
  30 X1(I)=PS(I)
    CALL EVAL(X1,YS,G1,W1)
    WRITE(6,520) ITER,YS
520 FORMAT(35HOSUM OF SQUARES OF FUNCTIONS AFTER ,15,15H  ITERATIONS =C69E0560
      1,E20.8)
  IF(YS.GE.YL) GO TO 50
  DO 31 I=1,IS
    PSS(I)=(1.+GAMMA)*PS(I)-GAMMA*PCENT(I)
  31 X1(I)=PSS(I)
    CALL EVAL(X1,YSS,G1,W1)
  40 CONTINUE
  IF(YSS.GE.YL) GO TO 55
  DO 43 I=1,IS
    PH(I)=PSS(I)
  GO TO 80
  50 CONTINUE
  DO 51 I=1,N
    IF(IX1.EQ.I) GO TO 51
  IF(YS.LT.Y(I)) GO TO 55
  51 CONTINUE
  GO TO 60
  55 CONTINUE
  DO 56 I=1,IS
    PH(I)=PS(I)
  GO TO 80
  60 CONTINUE
  IF(YS.GT.YH) GO TO 68

```

```
DO 61 I=1,IS
61 PH(I)=PS(I)
68 CONTINUE
DO 69 I=1,IS
69 PSS(I)=BETA*PH(I)+(1.-BETA)*PCENT(I)
70 CONTINUE
IF(YSS.GT.YH) GO TO 75
DO 71 I=1,IS
71 PH(I)=PSS(I)
75 CONTINUE
GO TO 80
DO 76 I=1,N
DO 76 J=1,IS
76 P(I,J)=(P(I,J)+PL(J))/2.
GO TO 5
80 CONTINUE
DO 81 I=1,IS
P(XI,I)=PH(I)
81 CONTINUE
GO TO 5
200 CONTINUE
DO 201 I=1,IS
201 X(I)=PL(I)
IER=0
RETURN
900 IER=3
RETURN
END
```


C69F0000
C69F0010
C69F0020
C69F0030
C69F0040
C69F0050
C69F0060
C69F0070
C69F0080

```
$IBFTC C69F      DECK
SUBROUTINE PRINT1(X)
COMMON /INIT/ IS,IN,DEL, EPS , METHOD,ITER
DIMENSION X(20)
WRITE(6,10) ITER, (X(I),I=1,IS)
10 FORMAT(1H1,45HSOLUTION TO THE SYSTEM INPUT WAS OBTAINED IN ,I3,
1      1H 11H ITERATIONS /, (E20.8) )
RETURN
END
```

LMSC/HREC A783333

```

$IBFTC C69G  DECK
SUBROUTINE CONTOUR
COMMON /CNTR/ XL,XR,YB,YT,N,SF
COMMON /INIT/ IS,IN,DEL,EPS,METHOD,ITER
COMMON /DATA/ F(1300),X(1300),Y(1300)
DIMENSION ITABLE(1250)
DIMENSION PAR(20,20),X1(20)
CALL CAMRAV(9)
C
WRITE(6,10) N,XL,XR,YB,YT
10 FORMAT(63H1FOR CONTOUR PLOTTING THE FOLLOWING OPTIONS HAVE BEEN SEC
1LECTED /22HNUMBER OF CONTOURS = ,13/19HOFOR GRID SELECTION/
2 5H XL =,E20.8/5H XR =,E20.8/5H YB =,E20.8/5H YT =,E20.8)
C
C
XV=XL
YV=YT
DX=(XR-XL)/35.
DY=(YT-YB)/35.
CALL GRID1V(1,XL,XR,YB,YT,DX,DY,-6,-6,-6,-3,-3)
DO 12 I=1,1250
12 ITABLE(I)=0
I1=1
I2=71
DO 15 K=1,36
XV = XL-DX
DO 13 I=I1,I2,2
X(I)=XV+DX
XV=X(I)
13 Y(I)=YV
YV=YV-DY
I1=I1+1
IF(2*((I1-1)/2) .EQ. (I1-1)) I1=I1+70
I2=I2+1
IF(2*((I2-1)/2) .EQ. (I2-1)) I2=I2+70
15 CONTINUE
DO 30 I=1,1296
X1(I)=X(I)
X1(2)=Y(I)
CALL EVAL(X1,F1,F2,PAR)
C69G0000
C69G0010
C69G0020
C69G0030
C69G0040
C69G0050
C69G0060
C69G0070
C69G0080
C69G0090
C69G0100
C69G0110
C69G0120
C69G0130
C69G0140
C69G0150
C69G0160
C69G0170
C69G0180
C69G0190
C69G0200
C69G0210
C69G0220
C69G0230
C69G0240
C69G0250
C69G0260
C69G0270
C69G0280
C69G0290
C69G0300
C69G0310
C69G0320
C69G0330
C69G0340
C69G0350
C69G0360
C69G0370
C69G0380
C69G0390

```

```

F(I)=F1
30 CONTINUE
FMIN=F(1)
FMAX=F(1)
DO 35 I=1,1296
IF(FMIN.GT.F(I)) FMIN=F(I)
IF(FMAX.LT.F(I)) FMAX=F(I)
35 CONTINUE
V=N-1
DF=(FMAX-FMIN)/V
IF (SF.NE.1.) DF = (FMAX-FMIN)/SF**(N-1)
WRITE(6,36) FMAX,FMIN,DF
36 FORMAT(8HOFMAX = ,E20.8/8H FMIN = ,E20.8/6H DF = ,E20.8)
I1=1
I2=4
IFLAG=1
J=1
DO 50 K=1,35
DO 45 I=1,35
V2=ABS(F(I1))-ABS(F(I2))
V1=ABS(F(I1+1))-ABS(F(I2-1))
ITABLE(J)=1
IF(V2.LT.V1) ITABLE(J)=2
J=J+1
I1=I1+2
45 I2=I2+2
GO TO (46,47),IFLAG
46 I1=I1-69
I2=I2+4
IFLAG=2
GO TO 50
47 I1=I1+1
I2=I2-69
IFLAG=1
50 CONTINUE
SFC = 1.
FC = FMIN - SF*DF
DO 200 I=1,N
IF (SF.NE.1.) GO TO 60
FC = FC + DF

```

```

GO TO 65
60 SFC = SFC*SF
FC = FMIN + SFC*DF
65 WRITE (6,1000) I,FC
1000 FORMAT (1H0,16H FOR CONTOUR NO. 13,18H THE VALUE OF F IS E16.8)
NS=1
IBX=1
IBOX=1
IBY=1
IFL2=0
I1=1
I2=2
109 IF (MID(FC,F(11),F(12)),GT.0) GO TO 150
110 IF (MID(FC,F(12),F(12+2)),GT.0) GO TO 155
111 IX=ITABLE(IBOX)
IF(IX.EQ. 1)I1=I1+2
IF(IX.EQ. 2)I2=I2+2
112 IF(IX.EQ. 1)I2=I2+2
IF(IX.EQ. 2)I1=I1+2
IF(IX.EQ. 35) GO TO 120
IBX=IBX+1
IBOX=IBOX+1
GO TO 109
120 IBX=1
IBY=IBY+1
IBOX=IBOX+1
IF(IBOX.GT. 1225) GO TO 200
IF(IFL2.EQ. 1) GO TO 121
I2=I2+1
I1=I1-69
IFL2=1
GO TO 109
121 I1=I1+1
I2=I2-69
IFL2=0
GO TO 110
150 XP = X(I1)
CALL LINTBL(YP,Y(I1),Y(I2),F(I1),F(I2),FC)
CALL POINTV(XP,YP,-NS)
GO TO 110

```

```

155 YP = Y(I2)
    CALL LINTBL(XP,X(I2),X(I2+2),F(I2),F(I2+2),FC)
    CALL POINTV(XP,YP,-NS)
    GO TO 111
200 CONTINUE
    CALL CLEAN
    RETURN
    END

```

C69G1200
 C69G1210
 C69G1220
 C69G1230
 C69G1240
 C69G1250
 C69G1260
 C69G1270

```

$IBFTC C69G      DECK
SUBROUTINE CONTOR
COMMON /CNTR/ XL,XR,YB,YT,N,SF
COMMON /INIT/ IS,IN,DEL,EPS,METHOD,ITER
COMMON /DATA/ F(1300),X(1300),Y(1300)
DIMENSION ITABLE(1250)
DIMENSION PAR(20,20),X1(20)
CALL CAMRAV(9)

C
WRITE(6,10) N,XL,XR,YB,YT
10 FORMAT(63H1FOR CONTOUR PLOTTING THE FOLLOWING OPTIONS HAVE BEEN SEC
11LECTED /22HNUMBER OF CONTOURS = ,13/19HOFOR GRID SELECTION/
2 5H XL =,E20.8/5H XR =,E20.8/5H YB =,E20.8/5H YT =,E20.8)

C
C
XV=XL
YV=YT
DX=(XR-XL)/35.
DY=(YT-YB)/35.
CALL GRIDIV(1,XL,XR,YB,YT,DX,DY,-6,-6,-6,-3,-3)
D0 12 I=1,1250
12 ITABLE(I)=0
I1=1
I2=71
D0 15 K=1,36
XV = XL-DX
D0 13 I=I1,I2,2
X(I)=XV+DX
XV=X(I)
13 Y(I)=YV
YV=YV-DY
I1=I1+1
IF(2*((I1-1)/2).EQ.(I1-1)) I1=I1+70
I2=I2+1
IF(2*((I2-1)/2).EQ.(I2-1)) I2=I2+70
15 CONTINUE
D0 30 I=1,1296
X1(I)=X(I)
X1(2)=Y(I)
CALL EVAL(X1,F1,F2,PAR)

```

```

F(I)=F1
30 CONTINUE
FMIN=F(I)
FMAX=F(I)
DO 35 I=1,1296
IF(FMIN.GT.F(I)) FMIN=F(I)
IF(FMAX.LT.F(I)) FMAX=F(I)
35 CONTINUE
V=N-1
DF=(FMAX-FMIN)/V
IF (SF.NE.1.) DF = (FMAX-FMIN)/SF**(N-1)
WRITE(6,36) FMAX,FMIN,DF
36 FORMAT(8HOFMAX = ,E20.8/8H FMIN = ,E20.8/6H DF = ,E20.8)
I1=1
I2=4
IFLAG=1
J=1
DO 50 K=1,35
DO 45 I=1,35
V2=ABS(F(I1))-ABS(F(I2))
V1=ABS(F(I1+1))-ABS(F(I2-1))
ITABLE(J)=1
IF(V2.LT.V1) ITABLE(J)=2
J=J+1
I1=I1+2
I2=I2+2
45 GO TO (46,47),IFLAG
46 I1=I1-69
I2=I2+4
IFLAG=2
GO TO 50
47 I1=I1+1
I2=I2-69
IFLAG=1
50 CONTINUE
SFC = 1./SF
FC = FMIN - DF
DO 200 I=1,N
IF (SF.NE.1.) GO TO 60
FC = FC + DF

```

```

GO TO 65
60 SFC = SFC*SF
FC = FMIN + SFC*DF
IF (I.EQ.1) FC = FMIN
65 WRITE (6,1000) I,FC
1000 FORMAT (1H0,16H FOR CONTOUR NO. I3,18H THE VALUE OF F IS E16.8)
NS=1
IBX=1
IBOX=1
IBY=1
IFL2=0
I1=1
I2=2
109 IF (MID(FC,F(I1),F(I2)).GT.0) GO TO 150
110 IF (MID(FC,F(I2),F(I2+2)).GT.0) GO TO 155
111 IX=ITABLE(IBOX)
IF (IX.EQ. 1) I1=I1+2
IF (IX.EQ. 2) I2=I2+2
112 IF (IX.EQ. 1) I2=I2+2
IF (IX.EQ. 2) I1=I1+2
IF (IBX.EQ. 35) GO TO 120
IBX=IBX+1
IBOX=IBOX+1
GO TO 109
120 IBX=1
IBY=IBY+1
IBOX=IBOX+1
IF (IBOX.GT. 1225) GO TO 200
IF (IFL2.EQ. 1) GO TO 121
I2=I2+1
I1=I1-69
IFL2=1
GO TO 109
121 I1=I1+1
I2=I2-69
IFL2=0
GO TO 110
150 XP = X(I1)
CALL LINTBL(YP,Y(I1),Y(I2),F(I1),F(I2),FC)
CALL POINTV(XP,YP,-NS)

```


C69G1200
C69G1210
C69G1220
C69G1230
C69G1240
C69G1250
C69G1260
C69G1270
C69G1280

GO TO 110
155 YP = Y(I2)
CALL LINTBL(XP,X(I2),X(I2+2),F(I2),F(I2+2),FC)
CALL POINTV(XP,YP,-NS)
GO TO 111
200 CONTINUE
CALL CLEAN
RETURN
END

C69H0000
C69H0010
C69H0020
C69H0030
C69H0040
C69H0050

\$IBFTC C69H DECK
SUBROUTINE LINTBL(R,LB,UB,FL,FU,F)
REAL LB
R = LB + (UB-LB)*(F-FL)/(FU-FL)
RETURN
END

C69I0000
C69I0010
C69I0020
C69I0030
C69I0040
C69I0050
C69I0060

\$IBFTC C69I DECK
FUNCTION MID(A,B,C)
MID = 0
IF (A.LT.B.AND.A.GT.C) MID = 1
IF (A.GT.B.AND.A.LT.C) MID = 1
RETURN
END

\$DATA

APPENDIX C-2 FALSE POSITION PROGRAM LISTING

C54E0001

* SUBROUTINE ZEROES, COMPUTES THE ROOT OF $F(X)=0$

* CALLING SEQUENCE TO SOLVE THE EQUATION $F(X,Y,\dots)=0$

* CALL ZEROES(X,XT,FXT,RATIO,PREC,XMIN,XMAX,6HTPPRR,F(X,Y,\dots))

* INDICATORS GUIDE

* *****

* 1) CODING BITS 26 TO 35

* BIT 26=1 ARITHMETIC TABULATION

* BIT 27=1 GEOMETRIC TABULATION

* BIT 28=1 INCREASING FUNCTION

* BIT 29=1 DECREASING FUNCTION

* BIT 30=1 INITIAL VALUE SUPERIOR TO THE ROOT

* BIT 31=1 INITIAL VALUE INFERIOR TO THE ROOT

* BIT 32=1 FUNCTION WITH AN UNDEFINED MODE OF VARIATION

* BIT 33=1 SUPERIOR VALUE

* BIT 34=1 INFERIOR VALUE

* BIT 35=1 BEST VALUE OF THE RESULT

* 2) LINKAGE TABUL BITS 15 TO 17

* TABUL =2 COMPUTATION OF $F(V)$ FOR $V=VT$

* TABUL =5 THE MODE OF VARIATION IS UNKNOWN. $X=XT+PAS$ OR $X=XT-PAS$

* TABUL =1 SEARCH OF THE BOUNDARIES

* TABUL =0 BOUNDARIES FOUND, SEARCH OF THE ROOT

* 3) LINKAGE AIGD BIT 14

* AIGD =1 INTERSECTION WITH A FIXED SLOPE LINE

* AIGD =0 DICHOTOMY

* 5) COUNTER BITS 7 AND 8

* ADDS 1 TO ITS CONTENT EACH TIME ONE ADDS A BIT TO ONE OF THE
* BOUNDARIES. THIS OPERATION CAN BE DONE ONLY 2 TIMES

* 6) FLAGS 1 AND 2 BITS 5 AND 4

* FLAG2 =1 IF XT IS GREATER THAN OR EQUAL TO $XMAX$

* FLAG1 =1 IF XT IS LESS THAN OR EQUAL TO $XMIN$

* 7) ERROR BIT BIT 3

* INDICATES THAT THE VARIABLE REACHED ONE OF THE BOUNDARIES

ENTRY ZERØS

***** MACRØ-INSTRUCTION GENERATING THE BLØCKS *****
***** ØF PARAMETERS RELATED WITH EACH STAGE *****

BLØCK	MACRØ	I
	IRP	I
RATE-I	BSS	1
IND-I	BSS	1
VI-I	BSS	1
FVI-I	BSS	1
VS-I	BSS	1
FVS-I	BSS	1
VA-I	BSS	1
FVA-I	BSS	1
V-I	BSS	1
MA-I	BSS	1
M-I	BSS	1
LKDR-I	PZE	
	BCI	1,ZERØS-I
CALL-I	TSX	**,4
	TRA	RET.AP
	PZE	,,LKDR-I
	BSS	N
	SPACE	2
	IRP	
	ENDM	BLØCK

***** PARAMETERS ØF DIMENSION *****

N	SET	3
NNN	SET	N+16
SIZE	BLØCK	(1,2,3)

RESTAURATION

ZERØS	TRA	STI	
	CLA	LKDR1,2	RESTAURATION ØF THE ADDRESS ØF INDEX 4
	STA	AXT4	
	TXI	**,2,NNN	
	SXD	STAGE,2	
X2	AXT	**,2	RESTAURATION ØF THE INDEXES
X1	AXT	**,1	
AXT4	AXT	**,4	
	LDI	PRØTI	PRØTECTION ØF THE INDICATOR REGISTER
	TRA	1,4	RETURN TØ THE CALLING PRØGRAM